ISSN 0252–9742

Bulletin

of the

European Association for

Theoretical Computer Science

EATCS

Number 143 June 2024

Council of the European Association for Theoretical Computer Science

President:	Artur Czumaj	United Kingdom
Vice Presidents:	Anca Muscholl	France
	Giuseppe F. Italiano	Italy
Treasurer:	Jean-Francois Raskin	Belgium
Bulletin Editor:	Stefan Schmid	Germany

Antoine Amarilli	France	Giuseppe F. Italiano	Italy
Ivona Bezakova	USA	Emanuela Merelli	Italy
Tiziana Calamoneri	Italy	Anca Muscholl	France
Thomas Colcombet	France	Charles Paperman	France
Artur Czumaj	UK	Tal Rabin	USA
Anne Driemel	Germany	Eva Rotenberg	Denmark
Javier Esparza	Germany	Jiri Sgall	Czech Republic
Gabriele Fici	Italy	Jukka Suomela	Finland
Inge Li Goertz	Denmark	Szymon Torunczyk	Poland
Fabrizio Grandoni	Switzerland	Bianca Truthe	Germany
Thore Husfeldt	Sweden, Denmark		

Past Presidents:

Maurice Nivat	(1972–1977)	Mike Paterson	(1977–1979)
Arto Salomaa	(1979–1985)	Grzegorz Rozenberg	(1985–1994)
Wilfred Brauer	(1994–1997)	Josep Díaz	(1997–2002)
Mogens Nielsen	(2002–2006)	Giorgio Ausiello	(2006–2009)
Burkhard Monien	(2009–2012)	Luca Aceto	(2012–2016)
Paul Spirakis	(2016–2020)		

Secretary Office:	Dmitry Chistikov	UK
	Efi Chita	Greece

EATCS Council Members

email addresses

Antoine Amarilli	a3nm@a3nm.net
Ivona Bezakova	ib@cs.rit.edu
Tiziana Calamoneri	calamo@di.uniroma1.it
Thomas Colcombet	Thomas.Colcombet@irif.fr
Artur Czumaj	A.Czumaj@warwick.ac.uk
Anne Driemel	driemel@cs.uni-bonn.de
Javier Esparza	esparza@in.tum.de
Gabriele Fici	gabriele.fici@unipa.it
Inge Li Goertz	inge@dtu.dk
Fabrizio Grandoni	fabrizio@idsia.ch
Thore Husfeldt	thore@itu.dk
Giuseppe F. Italiano	giuseppe.italiano@uniroma2.it
Emanuela Merelli	emanuela.merelli@unicam.it
Anca Muscholl	anca@labri.fr
Charles Paperman	charles.paperman@univ-lille.fr
Tal Rabin	chair.sigact@sigact.acm.org
Jean-Francois Raskin	jraskin@ulb.ac.be
Eva Rotenberg	eva@rotenberg.dk
Stefan Schmid	stefan.schmid@tu-berlin.de
Jiri Sgall	sgall@iuuk.mff.cuni.cz
Jukka Suomela	jukka.suomela@aalto.fi
Szymon Torunczyk	szymtor@mimuw.edu.pl
Bianca Truthe	bianca.truthe@informatik.uni-giessen.de

Bulletin Editor:	Stefan Schmid, Berlin, Germany
Cartoons:	DADARA, Amsterdam, The Netherlands

The bulletin is entirely typeset by pdfTEX and ConTEXt in TXfonts.

All contributions are to be sent electronically to

`bulletin@eatcs.org`

and must be prepared in LaTeX 2ε using the class `beatcs.cls` (a version of the standard LaTeX 2ε `article` class). All sources, including figures, and a reference PDF version must be bundled in a ZIP file.

Pictures are accepted in EPS, JPG, PNG, TIFF, MOV or, preferably, in PDF. Photographic reports from conferences must be arranged in ZIP files layed out according to the format described at the Bulletin's web site. Please, consult `http://www.eatcs.org/bulletin/howToSubmit.html`.

We regret we are unfortunately not able to accept submissions in other formats, or indeed submission not *strictly* adhering to the page and font layout set out in `beatcs.cls`. We shall also not be able to include contributions not typeset at camera-ready quality.

The details can be found at `http://www.eatcs.org/bulletin`, including class files, their documentation, and guidelines to deal with things such as pictures and overfull boxes. When in doubt, email `bulletin@eatcs.org`.

Deadlines for submissions of reports are January, May and September 15th, respectively for the February, June and October issues. Editorial decisions about submitted technical contributions will normally be made in 6/8 weeks. Accepted papers will appear in print as soon as possible thereafter.

The Editor welcomes proposals for surveys, tutorials, and thematic issues of the Bulletin dedicated to currently hot topics, as well as suggestions for new regular sections.

The EATCS home page is `http://www.eatcs.org`

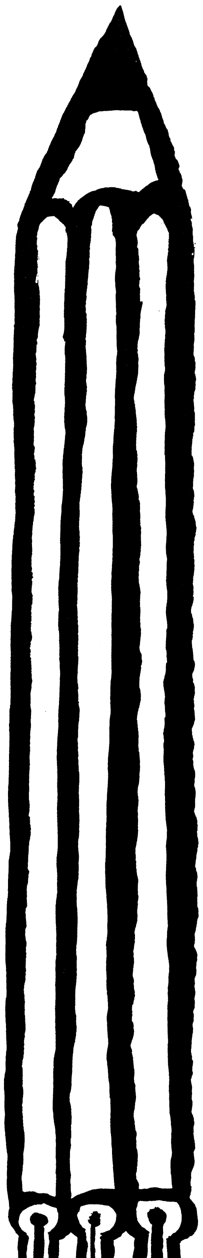
Table of Contents

EATCS Matters

Dear EATCS members,

As usual, the June issue of the Bulletin will be available just before ICALP, the flagship conference of the EATCS and an important meeting of the theoretical computer science community world-wide. The 51st ***EATCS International Colloquium on Automata, Languages, and Programming (ICALP 2023)****, will be held in Tallinn, Estonia, July 8-12, 2024 (https://compose.ioc.ee/icalp2024/). I am looking forward to the conference run again in a fully in-person mode. The PC chairs Karl Bringmann, Ola Svensson (track A), and Martin Grohe (track B), and the conference chair Pawel Sobocinski promise us an exciting scientific event. As usual, ICALP will be preceded by a series of workshops, which will take place before the conference. ICALP 2024 will be collocated with two further great events, LICS (39th Annual ACM/IEEE Symposium on Logic in Computer Science) and FSCD (9th International Conference on Formal Structures for Computation and Deduction). I look forward to the conference with you and I hope to see many of you during the ICALP 2024 conference in Tallinn!*

In the scientific part of the ICALP program, the Programme Committee chairs Karl Bringmann, Ola Svensson (track A), and Martin Grohe (track B), and their PCs have done fantastic job selecting an impressive collection of papers. The programme of ICALP 2024 will highlight research across many areas within theoretical computer science. I invite you to attend and/or watch the talks even outside your own



research field.

The **best paper awards at ICALP 2024** *will go to the following two papers:*

- *Track A:Yuda Feng and Shi Li, "A note on approximating weighted Nash social welfare with additive valuations;"*
- *Track B: Dmitry Chistikov, Alessio Mansutti and Mikhail Starchak, "Integer linear-exponential programming in NP by quantifier elimination."*

The **best student paper award** *(for a paper that is solely authored by a student) will go to the following three papers:*

- *Track A: Ce Jin and Hongxun Wu, "A faster algorithm for pigeonhole equal sums;"*
- *Track A: Kingsley Yung, "Limits of sequential local algorithms on the random k-XORSAT problem;"*
- *Track B: Roland Guttenberg, "Flattability of priority vector addition systems."*

Congratulations to the authors of the award-receiving papers!

In addition to regular research talks, ICALP 2024 will feature three invited talks delivered by

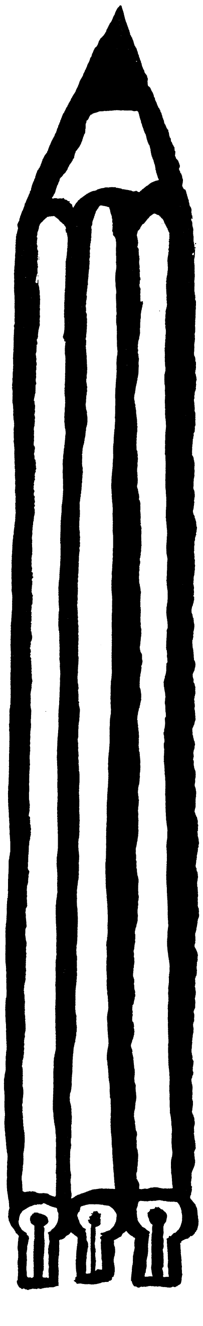
- *Anuj Dawar (University of Cambridge),*
- *Danupon Nanongkai (MPI Saarbrücken), and*
- *Merav Parter (Weizmann Institute),*

and further two joint (with LICS and FSCD) invited speakers

- *Edith Elkind (University of Oxford) and*
- *Stephanie Weirich (University of Pennsylvania).*

Apart from the invited and contributed talks, ICALP 2024 will feature special presentations of the receipts of the Gödel Prize 2024, EATCS Award 2024, the EATCS Presburger Award, and of the Alonzo Church Award:

- *Each year, the Gödel Prize (https://eatcs.org/index.php/goedel-prize) recognizes standout papers in theoretical computer science. The* ***Gödel Prize 2024*** *(sponsored jointly by the EATCS and the ACM SIGACT) is awarded to* ***Ryan Williams*** *(MIT) for presenting a novel paradigm for a "rich two-way connection" between algorithmic techniques and lower-bound methods.*
- *The EATCS annually honors a respected scientist from our community with the* ***EATCS Distinguished Achievements Award****, to acknowledge extensive and widely recognized contributions to theoretical computer science over a life long scientific career (http://eatcs.org/index.php/eatcs-award). The recipient of the* ***EATCS Award 2024*** *is* ***Samson Abramsky*** *(University College London), for his numerous fundamental contributions over the past four decades in the area of logic and semantics of computation.*
- *The* ***EATCS Presburger Award*** *(https://eatcs.org/index.php/presburger) is awarded by the EATCS to a young scientist for outstanding contributions*



in theoretical computer science, documented by a published paper or a series of published papers. The recipients of the **Presburger Award 2024** are jointly **Justin Hsu** and **Pravesh Kothari**.

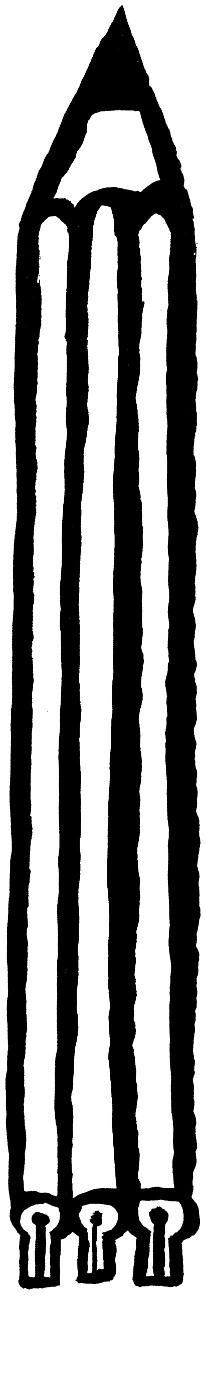
- The **Alonzo Church Award** for Outstanding Contributions to Logic and Computation (https://eatcs.org/index.php/church-award) is a major prize in cooperation of the EATCS with ACM Special Interest Group on Logic and Computation and the European Association for Computer Science Logic. The recipients of the **2024 Alonzo Church Award** are jointly **Thomas Ehrhard** and **Laurent Regnier**, for the introduction of the Differential λ-Calculus and Differential Linear Logic.

Moreover, during the conference, we will honor the recipients of the 2023 EATCS Distinguished Dissertation Award and the new group of EATCS Fellows.

The EATCS established the **EATCS Distinguished Dissertation Award** to promote and recognize outstanding dissertations in the field of Theoretical Computer Science (https://eatcs.org/index.php/dissertation-award). The three recipients of the **2023 EATCS Distinguished Dissertation Award** for the PhD dissertation in the field of Theoretical Computer Science that have been successfully defended in 2023 are

- **William Kuszmaul**: "Randomized Data Structures: New Perspectives and Hidden Surprises" (PhD at MIT; advisor: Charles E. Leiserson);

- **Nathan Klein**: "Finding Structure in Entropy: Improved Approximation



Algorithms for TSP and other Graph Problems" (PhD at the University of Washington; advisors: Anna Karlin and Shayan Oveis Gharan)

- ***Ruiwen Dong**: "Algorithmic Problems for Subsemigroups of Infinite Groups" (PhD at the University of Oxford; advisors: Christoph Haase and James Worrell).*

*The **EATCS Fellows** Program is established by the EATCS to recognize outstanding EATCS Members for their scientific achievements in the field of Theoretical Computer Science. The new group of **EATCS Fellows (class 2024)** consists of*

- ***Yossi Azar** (Tel-Aviv University, Israel) for "many seminal contributions to the study of online and approximation algorithms, and for his long-standing service to the community," and*
- ***Friedhelm Meyer auf der Heide** (Paderborn University, Germany) for "influential contributions to algorithmic and complexity-theoretic problems in parallel computing, communication and data management in networks, network dynamics, algorithms in computer graphics, and probabilistic analysis."*

In addition to the standard conference program, ICALP 2024 will also have ten satellite workshops co-located with the main conference, taking place on July 6-9, 2024, before the main event:

- *Algorithmic Aspects of Temporal Graphs VII (AATG 2024; July 7)*

- *Geometric and Topological Methods in Computer Science (GETCO 2024; July 6-7)*
- *Intersection Types and Related Systems (ITRS 2024; July 9)*
- *International Workshop on Confluence (IWC 2024; July 9)*
- *Learning and Automata (LearnAut 2024; July 7)*
- *Logical Frameworks and Meta-Languages: Theory and Practice (LFMTP 2024; July 8)*
- *Logic Mentoring Workshop (LMW 2024; July 7)*
- *Mathematically Structured Functional Programming (MSFP 202; July 8)*
- *Parameterized Approximation Algorithms Workshop (PAAW 2024; July 6)*
- *Parameterized Algorithms and Constraint Satisfaction (PACS 2024; July 7)*
- *Structure meets Power (SmP 2024; July 7)*
- *Trends in Arithmetic Theories (TAT 2024; July 6)*
- *Eighth International Workshop on Trends in Linear Logic and Applications (TLLA 2024; July 8-9)*
- *Women in Logic 2024 (9 July).*

Congratulations to the award winners and new EATCS Fellows!

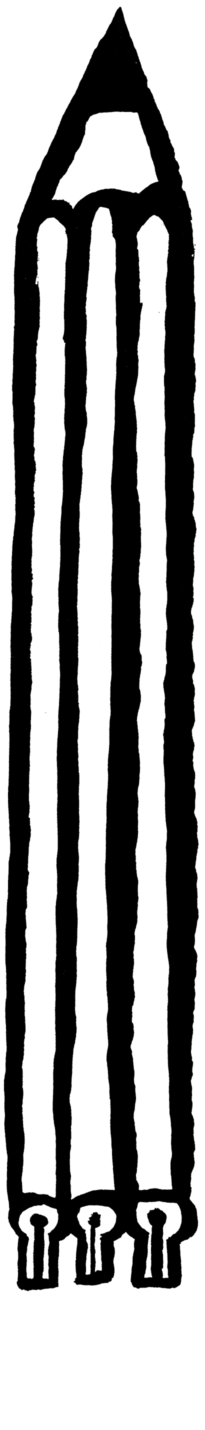
On behalf of the EATCS, I also heartily thank the members of the awards,

dissertation and fellow committees for their work in the selection of this stellar set of award recipients and fellows. It will be a great honor to celebrate the work of these colleagues during ICALP 2024. (More details about the EATCS Award 2024, the EATCS Presburger Award 2024, 2024 Alonzo Church Award, the 2023 EATCS Distinguished Dissertation Award, the EATCS Fellows, and the Gödel Prize 2024 are presented on the later pages of this issue of the Bulletin.)

Also, please allow me to remind you about three other EATCS affiliated conferences that will be taking place later this year.

- ***MFCS 2024:*** *the 49th International Symposium on Mathematical Foundations of Computer Science, will be held in Bratislava, Slovakia, August 26-30, 2024 (http://www.mfcs.sk/).*
- ***ESA 2024:*** *the 32nd Annual European Symposium on Algorithms, will be held at Royal Holloway, University of London in Egham, United Kingdom, September 2-4, 2024 (https://algo-conference.org/2024/esa/).*
- ***DISC 2024:*** *the 38th International Symposium on Distributed Computing, will be held in Madrid, Spain, October 28 - November 1, 2024 (http://www.disc-conference.org/wp/disc2024/).*

As usual, let me close this letter by reminding you that you are always most welcome to send me your comments, criticisms and suggestions for improving the impact of the EATCS on the Theoretical Computer Science community at

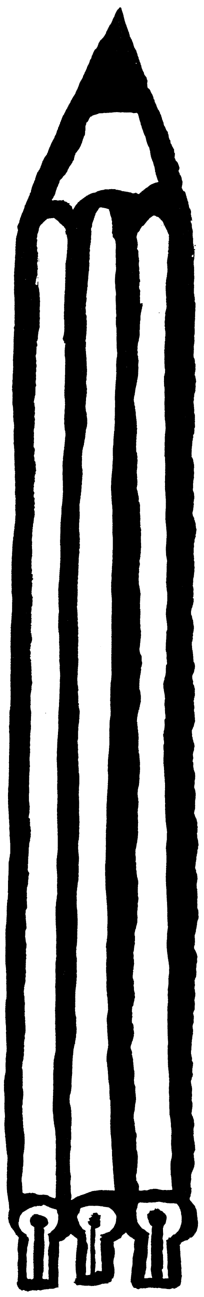


president@eatcs.org. We will consider all your suggestions and criticisms carefully.

I look forward to seeing many of you at ICALP 2024 and to discussing ways of improving the impact of the EATCS within the theoretical computer science community at the general assembly.

Artur Czumaj
University of Warwick, UK
President of EATCS
president@eatcs.org

June 2024



Dear EATCS member!

In front of you is the Summer issue of the Bulletin.

The Interview Column features Sergio Rajsbaum. He tells us about his experiences when, 30 years ago, he travelled all the way from Mexico to Israel to study at the Technion, and soon after became part of Shimon Even's academic family. He also reflects on what makes a good team mate, a good PhD advisor, and on the role of trust in general. The

Distributed Computing Column, edited by Seth Gilbert, features an article by Helen Xu who shares with us "everything we need to know" about how to design efficient parallel applications that operate on dynamic graphs, from container designs (i.e., data structures), over interfaces for interacting with a dynamic graph, to performance benchmarking. In the Formal

Language Theory Column, edited by Giovanni Pighizzini, Markus L. Schmid highlights the relationship between a recent research area in database theory, namely the information extraction framework of document spanners, and classical formal language theory. In

the Logic in Computer Science Column, Yuri Gurevich and Andreas Blass discuss some of the many foundational problems raised by the ongoing AI revolution, in a thought-provoking dialogue. The Bulletin

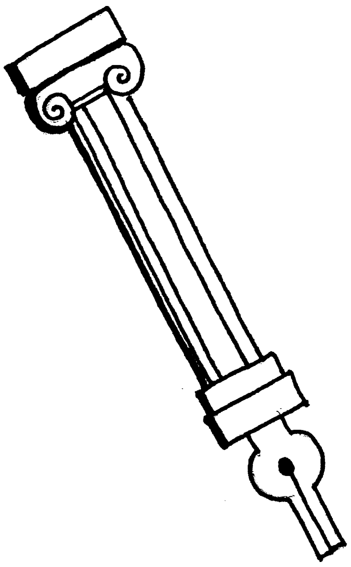
further includes reports from the EATCS Japan Chapter and the 40th British Colloquium for Theoretical Computer Science.

I wish you an inspiring read and a wonderful Summer!

Stefan Schmid, Berlin
June 2024

Institutional Sponsors

EATCS Columns



The Interview Column

BY

Chen Avin and Stefan Schmid
Ben Gurion University, Israel and TU Berlin, Germany
`{chenavin,schmiste}@gmail.com`

Know the Person behind the Papers

Today: Sergio Rajsbaum

Bio: *Sergio Rajsbaum received a degree in Computer Engineering from the Universidad Nacional Autónoma de México (UNAM) in 1985, and a PhD in the Computer Science from the Technion, Israel, in 1991. Since then he has been a faculty member at the Instituto de Matemáticas at UNAM. He did a postdoc at MIT, and is currently visiting the Institut de Recherche en Informatique Fondamentale in Paris. His research interests are in the theory of distributed computing. He is one of the world leading researchers on the topology approach to distributed computability, and published a book on the subject with Maurice Herlihy and Dmitry Kozlov. He has been chair of the Program Committee of conferences such as LATIN, PODC, SIROCCO and a member of the editorial board of IEEE Transactions on Dependable and Secure Computing, Information Processing Letters, and Computer Science Review.*

EATCS: Do you have any advice for young researchers? In what should they invest time, what should they avoid?

SR: I once heard a talk by Yehuda Afek from Tel-Aviv University, where he was describing his experience with creating a startup company. He explained that there are many great ideas out there, that the key to succeeding in the face of competition to other proposals with good ideas, was that he had put together a team of extraordinary people. My advise to young researchers is to always be close to colleagues which are excellent both academically and as teammates.

EATCS: What do you mean by being excellent as teammate?

SR: This is a great opportunity to honor my PhD advisor Shimon Even (1935–2004). In addition to his pioneering research contributions and his role in establishing computer science education in Israel, he is known for having been a highly influential educator. I believe being such a good educator is related to being a good teammate. To feel deeply that that you and your student belong to the same team.

EATCS: This seems related to the topic of being a good PhD advisor.

SR: Certainly, Shimon was for me a role model, a professional inspiration, in a broad sense, an "academic father," and I think a reason for that is a feeling that I

still remember more that 30 years later, when I first met him: I was entering his "academic family." This had various, diverse implications. I experienced for the first time the feeling of someone listening to me with full attention, with great interest. I have experience since then that great researchers are also great listeners. What made Shimon special is that this was the case in both our research discussions and outside of them. When I was talking about a new idea, about how to prove a result, he always payed full attention. He was equally interested about topics outside academia, and about myself, a young student coming all the way from Mexico to study in Israel.

EATCS: What does it take for someone to be able to be a great listener?

SR: I think a fundamental component needed to achieve this level of interest and attention for a colleague in a team, is *trust*. I am not sure why, but Shimon had full trust in me academically, and as a person. I was not a particularly good student, of course compared to the top level students of the Technion, and in particular with his former PhD students, just to mention the two that came before me, world stars such as Oded Goldreich (1983) and Baruch Awerbuch (1984). But additionally I came from Mexico with an average level of undergraduate education, and struggling with the lectures which were then all in Hebrew.

EATCS: Is there a nice anecdote you would like to share with our readers from the days of your PhD?

SR: I saw for myself how *trust* can change the life of a person. After taking a few courses as a Masters student at the Technion, I needed to find an advisor. I loved graph theory, and Shimon was famous for his book on graph algorithms, so I went to his office and told him I was looking for an advisor, without knowing much more about him. He received me happily, also without knowing much about me. I still remember the long conversation, he was a brilliant conversationalist. He told fun anecdotes about famous researchers, asked me about Mexico, and gave me a paper to read, and soon after he had accepted me as a student. Immediately my status changed at the Technion, all secretaries where nice to me, and I became famous with the students, which asked me how I dared to knock the door of such a famous professor, who in addition was feared by many. He was imposing physically, spoke loudly and was very opinionated.

EATCS: We ask all interviewees to share a photo with us. Can you please tell us a little bit more about the photo you shared?

SR: Becoming part of Shimon's academic family included having fun together. This I have found over the years, is important for me to do good research, both during the discussion sessions, and around it, in long walks, going out for dinner, etc. The picture is while traveling in France, we rented a car and visited beautiful villages to get to a conference where we were presenting a paper. Such experiences contributed to making him into such an influential figure for me. Being a scientist for me became a highly social endeavor, that includes sharing interests in culture, in traveling, eating, but most of all, as Shimon would say, talking. My love for Wagner is due to his taking me to Wagner's opera The Flying Dutchman at The Met, in a trip to present a paper in New York, where for 2.5 hours my ears heard nothing but noise.

EATCS: What makes someone a good advisor?

SR: Indeed, I share the feeling expressed by Oded, “he also fits in the set of dozen people (including my parents...) that have most influenced my life at large.” and as him, after 30 years, I still find myself repeating “my adviser, Shimon Even, would say....". He had very strong opinions, and in particular, a very clear sense of what is good research and what is not; he would say that when a problem looks natural to him, he does not need to see an application. He would tell me never to cite a theorem that I don’t understand, and instead of going straight to read the proof, always try to prove it myself.

EATCS: Is there a paper which influenced you particularly, and which you recommend other community members to read?

SR: An advise from Shimon was to follow the papers of great researchers, he told me, read Awerbuch’s paper “Complexity of network synchronization" (JACM 1985), and explain it to me. This way of working with a student can be very effective, and indeed this paper was my introduction to the world of scientific research, and to distributed computing: learning about synchronous versus asynchronous computing models, about how one can simulate one model on top of another. A related beautiful paper by Shimon that I would recommend is “Marked Directed Graphs" (J. Comput. Syst. Sci. 1971), about a special class of Petri Nets.

EATCS: Is there a paper of your own you like to recommend the readers to study? What is the story behind this paper?

SR: When I presented Awerbuch’s paper to Shimon, he noticed that there was an interesting issue with the initialization of the computation, and we began to suspect that there was no need for a special routine that would start all processes at the same round, as in a Firing Squad solution (this is how I learned about this beautiful problem). My first research result was that the processes would synchronize themselves naturally, and we gave to such a process the name *unison* starting a research area that is still active today, popularized by a paper on self-stabilizig unison by Mohamed Gouda and Ted Herman (IPL 1990). This was also the beginning of a long list of additional outstanding teammates, “angels” that have appeared along my academic life, trusted me and to which I thank for my career development. Soon after publishing our unison paper (Seq. Comb. Comp. Sec. and Transm., Springer 1990) Paul Spirakis invited me to an unforgettable stay in Patras; Hagit Attiya arrived at the Technion after her posdoc and during my last year as a PhD at the Technion; Amir Herzberg with whom I shared an office, and then Boaz Patt-Shamir, with the paper I would recommend readers, “A theory of clock synchronization" (STOC 1994), a simple addition of timing constrains to the events of Lamport’s causal order.

EATCS: What is your most important scientific contribution?

SR: I am not sure, but continuing with the list of the first angels that have trusted me, accompanied me and made a pleasure being in academics all these years, there is Maurice Herlihy, arriving to my life the same year I was a posdoc at MIT with Nancy Lynch and working with Boaz, and introducing me to the topology approach to distributed computability, this year we are celebrating the 30th anniversary of our first publication on the topic (PODC 1994).

The Formal Language Theory Column

by

Giovanni Pighizzini

Dipartimento di Informatica
Università degli Studi di Milano
20133 Milano, Italy
pighizzini@di.unimi.it

A Short Note on the Mutually Beneficial Relationship Between Information Extraction in Database Theory and Classical Formal Language Theory

Markus L. Schmid
Humboldt Universität zu Berlin, Berlin, Germany
MLSchmid@MLSchmid.de

Abstract

This short note highlights the relationship between a recent research area in database theory, namely the information extraction framework of document spanners, and classical formal language theory.

1 Formal Languages in Other Areas of TCS

Formal language theory is one of the main research areas of theoretical computer science, but its importance and relevance has changed (although not declined) over time. For example, Hopcroft, Motwani and Ullman state in their famous textbook [25] that "in 1979, automata and language theory was still an area of active research", while "today, there is little direct research in automata theory", where *today* refers to 2007. This assessment by Hopcroft, Motwani and Ullman, which they gave mainly as a justification for changing the focus and content of their textbook, has offended some of the purists among the formal languages researchers.

However, for young researchers in formal language theory who try to build scientific careers in the current landscape of theoretical computer science, it is perhaps a good idea to be less emotional and more pragmatic about these things. Neither abandoning formal languages altogether nor pretending that we are still in the 60s or 70s seems the right way to go. In this regard, it is interesting to note that Hopcroft, Motwani and Ullman also say in their preface that there is little direct research in automata theory "*as opposed to its applications*". In general, the preface of their book stresses the fact that the field of formal languages has grown from something quite special into standard content of undergraduate computer science courses, which is a good thing.

I do not want to take sides in the controversy of whether formal languages have become less important nowadays in research and teaching. But compared to roughly 60 years ago, it seems that formal language theory exists now less as a coherent and well-defined research area, but instead is scattered throughout the various sub-areas of theoretical computer science. This can be a great opportunity, since it means that various TCS-communities are interested in formal languages, whether they realise it or not. For researchers in formal languages, it is therefore beneficial to discover those "hidden gems" of formal language theory that can be found in seemingly unconnected areas of theoretical computer science.

One such area is information extraction in database theory, which shall be discussed in the next section. Note that it is not the purpose of this article to provide a survey of the field of information extraction, but rather to explain the connections between this topic and the area of formal languages.

2 A Formal Framework for Information Extraction

The concept of information extraction in database theory and the framework of document spanners is surveyed in more detail in [37, 2, 34]. Here, we focus on the relation between this area and classical formal language theory.

Document spanners (or simply spanners) have been introduced in the seminal paper [13], and they constitute a framework for extracting information from texts (i. e., strings, sequences or words, or, as is the common term in the data management community, documents); it is therefore called an *information extraction framework*. Since its introduction, many papers in database theory have been published that are concerned with document spanners (see [13, 18, 31, 3, 27, 14, 32, 15, 16, 19, 20, 29, 30, 12, 35, 36, 38, 28, 10, 8, 11, 4]). From a formal languages point of view, this framework is appealing, since it is a query mechanism that solely works on textual data.

A document spanner (over a set $\mathcal{X}$ of variables)[1] is a function that maps a word w to a finite table with a column for each variable from $\mathcal{X}$ and the entries of the cells of the table are just pairs of positions of w. This can be illustrated as follows:

$w = \mathsf{abbabccabc} \quad \Longrightarrow$

x	y	z
$(2,5)$	$(4,7)$	$(1,10)$
$(3,5)$	$(5,8)$	$(4,7)$
$(1,3)$	$(3,10)$	$(2,4)$
$\vdots$	$\vdots$	$\vdots$

[1]Let us fix $\mathcal{X} = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$ in our examples.

The pairs (i, j) are called *spans* and are interpreted as pointers to factors of w, e. g., $(4, 7)$ refers to $\mathtt{abcc}$, since this is the factor that starts at position 4 and ends at position 7. A row of the table is called a *span tuple*, the whole table is called a *span relation* and, as already mentioned above, a function S that maps each $w \in \Sigma^*$ to a span relation $S(w)$ is called a *document spanner*.[2] Regarding the question why spanners are a good formalisation of relevant information extraction tasks, the reader is referred to the surveys [37, 2, 34], or the introductions of the papers mentioned above.

A spanner S over variables $\mathcal{X}$ is therefore a function from Σ^* to the set of span relations over $\mathcal{X}$, but it can conveniently be represented as a formal language over the alphabet $\Sigma \cup \{{}^{\mathsf{x}}\triangleright, \triangleleft^{\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}\}$: We simply encode every span tuple t of the span relation $S(w)$ by enclosing the factor of w that corresponds to the span of $\mathsf{x} \in \mathcal{X}$ by the special marker symbols ${}^{\mathsf{x}}\triangleright$ and $\triangleleft^{\mathsf{x}}$ (we may think of these markers as pairs of brackets). For example, marking in $\mathtt{abbabccabc}$ the factor from position 2 to position 5 with ${}^{\mathsf{x}}\triangleright$ and $\triangleleft^{\mathsf{x}}$, the factor from position 4 to position 7 with ${}^{\mathsf{y}}\triangleright$ and $\triangleleft^{\mathsf{y}}$ and so on yields the string ${}^{\mathsf{z}}\triangleright\, \mathtt{a}\, {}^{\mathsf{x}}\triangleright\, \mathtt{bb}\, {}^{\mathsf{y}}\triangleright\, \mathtt{ab}\, \triangleleft^{\mathsf{x}}\, \mathtt{cc}\, \triangleleft^{\mathsf{y}}\, \mathtt{abc}\triangleleft^{\mathsf{z}}$, which encodes the span tuple $((2, 5), (4, 7), (1, 10))$ of the span relation $S(\mathtt{abbabccabc})$. We shall denote such marked strings as *subword-marked words*.[3]

In this way, any span relation $S(w)$ yields a finite set of subword-marked words and joining all these sets of subword-marked words yields a (usually infinite) language of subword-marked words that represents the spanner. Conversely, any language L of subword-marked words uniquely describes a spanner $[\![L]\!]$: For any word $w \in \Sigma^*$, the span relation $[\![L]\!](w)$ simply contains all span tuples described by some subword-marked word from L whose Σ-part equals w. For example, having ${}^{\mathsf{x}}\triangleright\, {}^{\mathsf{y}}\triangleright\, \mathtt{a} \triangleleft^{\mathsf{y}}\, \mathtt{b} \triangleleft^{\mathsf{x}}\, \mathtt{ab}\, {}^{\mathsf{z}}\triangleright\, \mathtt{c}\triangleleft^{\mathsf{z}}$ in L simply means that $[\![L]\!](\mathtt{ababc})$ contains the span tuple $((1, 2), (1, 1), (5, 5))$.[4]

A consequence of this relation between spanners and subword-marked languages is that spanners can be represented by formal language description mechanisms, like automata, expressions, grammars, or just any decision algorithm that accepts a subword-marked language. In particular, we can use those language description mechanisms as tools for specifying spanners as well as algorithmic tools for dealing with them computationally. For example, a spanner that produces a table of all pairs of a non-empty unary factor over the symbol $\mathtt{a}$ and a non-empty

[2]Here and henceforth in this article, we use Σ as our underlying terminal alphabet.

[3]The term *ref-word* is also common in the literature on spanners.

[4]We ignore the fact here that the representation of a spanner by means of a subword-marked language is not unique, since consecutive occurrences of marker symbols can be reordered without changing the described spanner. For example, ${}^{\mathsf{x}}\triangleright\, {}^{\mathsf{y}}\triangleright\, \mathtt{a} \triangleleft^{\mathsf{y}} \mathtt{b} \triangleleft^{\mathsf{x}} \mathtt{ab}\, {}^{\mathsf{z}}\triangleright\, \mathtt{c} \triangleleft^{\mathsf{z}}$ and ${}^{\mathsf{y}}\triangleright\, {}^{\mathsf{x}}\triangleright\, \mathtt{a} \triangleleft^{\mathsf{y}} \mathtt{b} \triangleleft^{\mathsf{x}} \mathtt{ab}\, {}^{\mathsf{z}}\triangleright\, \mathtt{c} \triangleleft^{\mathsf{z}}$ are different subword-marked words that nevertheless encode the same string and span tuple. This is a bit annoying and can cause some difficulties, but this aspect is thoroughly discussed and addressed in the literature on document spanners.

unary factor over the symbol b can be specified as the subword-marked language $\{u \,{}^{\mathsf{x}}\!\rhd\, c^n \lhd^{\mathsf{x}}\, v \,{}^{\mathsf{y}}\!\rhd\, d^m \lhd^{\mathsf{y}}\, w \mid u, v, w \in \Sigma^*, n, m \geq 1, \{c, d\} = \{\mathsf{a}, \mathsf{b}\}\}$. The subword-marked language $\{u \,{}^{\mathsf{x}}\!\rhd\, c^n \lhd^{\mathsf{x}}\, v \,{}^{\mathsf{y}}\!\rhd\, d^n \lhd^{\mathsf{y}}\, w \mid u, v, w \in \Sigma^*, n \geq 1, \{c, d\} = \{\mathsf{a}, \mathsf{b}\}\}$ describes the variant of the spanner, where the two factors must have the same length.

It is probably not surprising that *regular* spanners, i. e., those spanners that can be represented by regular subword-marked languages, play a central role in the area of document spanners. Such spanners can be represented by regular expressions like $\mathsf{a} \,{}^{\mathsf{x}}\!\rhd\, (\mathsf{b} + \mathsf{c})^* \,\mathsf{a} \lhd^{\mathsf{x}}\, \mathsf{b}^* \,{}^{\mathsf{y}}\!\rhd\, \mathsf{c}^* \lhd^{\mathsf{y}}\, \mathsf{c}^*$ (which is very convenient on the specification level) and as finite automata (which is useful for algorithms). In fact, regular spanners exhibit many desirable algorithmic properties, which follow from the good algorithmic properties of regular languages. On the other hand, the spanner mentioned above that extracts pairs of unary factors of the same size is obviously not a regular spanner (by the typical pumping argument). Obviously, we can consider arbitrary classes of subword-marked languages, which then describe classes of document spanners. An obvious choice that comes to mind are context-free document spanners, which have been investigated in the literature on information extraction (see [30, 29, 4]).

2.1 Research Tasks Motivated by Document Spanners

From a purely formal language theoretical point of view, it is interesting that document spanners – a recent hot topic in foundational database research – can be phrased in terms of classical (regular) languages. This means that we can use the known classical results about regular languages for regular spanners (and for their possible implementations). Probably more interesting is the fact that this information extraction perspective towards regular languages has also lead to new questions and computational problems about finite automata and formal languages in general that have not yet been addressed by the formal languages research of the last six decades. Let us now discuss some of these research questions.

Enumerating the Span Tuples: In the context of information extraction, it is an important task to enumerate for a given word w and a regular spanner S (given as finite automaton) all span tuples of $S(w)$ without repetition. Ideally, such an algorithm has a preprocessing phase that is linear in $|w|$ and then provides an enumeration with a delay (i. e., the time between two consecutive output elements) that is independent from $|w|$ (although it can still depend on the automaton for S). These requirements are justified by the practically relevant assumption that w represents data and is therefore very large, while S is a human-readable query and is therefore in comparison rather small. So these time bounds can be considered to be the optimum (assuming that we have to read the data at least once).

This is a problem on classical finite automata, but it has not yet been considered in the research of automata theory. Note that this problem is quite different from enumerating the words of a regular language, which has been investigated in formal language theory (see, e. g., [1]). It is also different from enumerating all accepting runs of an automaton on a given input word (i. e., paths between two nodes in a DAG), since such runs have length $|w|$, which would result in a prohibitively large delay. What we actually have to do is the following: For a given word $w \in \Sigma^*$, we have to consider all marked version of w (i. e., subword-marked words with a Σ-portion that equals w) that are accepted by the automaton, but for these marked versions, we only want to produce the marked positions as output (in the right format of course). Another way of phrasing this is in terms of finite transducers: We get a transducer that maps a string to a marked version of it (i. e., we have a finite set Γ of markers and the transducer has the choice of either marking an input symbol $\mathtt{a} \in \Sigma$ as $\mathtt{a}_\gamma$ for some $\gamma \in \Gamma$, or to leave it unmarked). Then, for a given input, we want to enumerate all possible marked outputs, but these outputs are to be represented only by the marked symbols along with their positions in the input string.

In any case, we are dealing with an algorithmic problem about classical finite automata, that is highly relevant in the context of document spanners, but does not seem to be covered by the classical studies on finite automata. An important particularity is the requirement that after the preprocessing we cannot spend time between two outputs that depends on the size of the input or the size of the accepting run.

As a main result in the field of document spanners, it has been shown that we can solve this enumeration problem with preprocessing linear in $|w|$ and delay independent of $|w|$ (see [3, 2, 14]).

An obvious next step is the question whether we can achieve the same for larger classes of spanners. Recall the spanner from above that is represented by the subword-marked language $\{u \, {}^{\mathsf{x}}\!\rhd \, c^n \lhd^{\mathsf{x}} \, v \, {}^{\mathsf{y}}\!\rhd \, d^n \lhd^{\mathsf{y}} \, w \mid u, v, w \in \Sigma^*, n \geq 1, \{c, d\} = \{\mathtt{a}, \mathtt{b}\}\}$. It is not difficult to see that this language is context-free. So can we also enumerate the span tuples of span relations extracted by context-free spanners (i. e., the class of spanners that can be described by context-free subword-marked languages)? It has been shown in [4] that for unambiguous grammars this is indeed possible with preprocessing that is cubic in $|w|$ and a delay that is independent of $|w|$ (observe that context-free parsing reduces to this problem, so the cubic bound seems to be necessary at least for combinatorial algorithms).

String Equality Selection and Angluin's Pattern Languages: In the literature on document spanners, a string equality selection operator has been introduced that simply removes those span tuples from a span relation for which certain spans *do not* refer to equal factors. It is used on top of a regular spanner to describe spanners that are non-regular due to equality checking of factors. For example, in this way we can describe spanners of the form $\{u\,\mathsf{a}\ {}^{\mathsf{x}}\!\triangleright\ v\ \triangleleft^{\mathsf{x}}\ u'\,\mathsf{b}\ {}^{\mathsf{y}}\!\triangleright\ v\ \triangleleft^{\mathsf{y}}\ u'' \mid u, u', u'', v \in \Sigma^*\}$, so a spanner that extracts two different occurrences of the same factor, the first one directly preceded by symbol a and the second one directly preceded by symbol b.

This string equality selection operator does not serve the mere purpose of playing around with the concept of document spanners, it is in fact already included in the original paper [13] and vital for covering the core of IBM's SystemT, which is the practical counterpart of the theoretical concept of document spanners.

What is interesting from a formal languages point of view is that the string equality selection operator turns spanners into a model that covers Angluin's pattern languages [6] and that is also related to regular expressions with backreferences [9, 33, 17]. Recall that Angluin's patterns are strings of the form $\mathsf{xabxaaay}$, where x and y are variables that can be substituted with arbitrary strings over Σ such that this pattern describes the pattern language $\{u\mathsf{ab}u\mathsf{aaa}v \mid u, v \in \Sigma^*\}$. Regular expressions with backreferences are regular expressions that can enclose a subexpression with special brackets ${}^{\mathsf{x}}\!\triangleright\ r \triangleleft^{\mathsf{x}}$ and then use variable x as a repetition of whatever was matched to r, e. g., $\mathsf{a}^*\ {}^{\mathsf{x}}\!\triangleright\ (\mathsf{a} + \mathsf{b})^*\ \triangleleft^{\mathsf{x}}\ \mathsf{c}^*(\mathsf{x}\,\mathsf{a} + (\mathsf{b}\,\mathsf{x})^*)$ is a regular expression with backreferences.

As a result of these connections to established classes of formal languages, many lower bounds for regular spanners extended with string equality selection can be obtained directly from known results, or by extending techniques used for pattern languages and regular expressions with backreferences (see [16]). Moreover, techniques previously used for regular expressions with backreferences have been applied to define a subclass of regular spanners with string equality selection that exhibits better decidability and complexity (see [35]).

Document Spanners on Compressed Input Strings: When dealing with strings, it is a classical and practically highly relevant question whether we can also handle the case where the input strings are compressed by the lossless compression scheme of so-called *straight-line programs* or SLPs for short (i. e., strings compressed by a context-free grammar which can derive only this string).

It is a comparatively simple observation that checking $w \in L(M)$ for an NFA M and a word w that is represented by an SLP G can be done in time linear in $|G|$ (and cubic in $|M|$). However, in terms of document spanners, this problems presents itself in a new light, i. e., as the extension to the more difficult (and arguably more interesting) problem of how to enumerate all span tuples of $S(w)$ for some regular spanner S in the case that w is given as an SLP G. It turns out that we can do this with a preprocessing that is only linear in $|G|$ and a delay that still only depends on $|M|$ (see [36, 28]). On the one hand, this result is a contribution to the field of SLP-compressed algorithmics (further demonstrating its usefulness), while, on the other hand, it further demonstrates the practical relevance of regular spanners.

In query evaluation, the dynamic setting is quite important: If we have enumerated the answers to a query over some data and then update our data by only changing a small portion of it, can we exploit this redundancy somehow such that when we want to enumerate our query again, we do not have to completely re-run the preprocessing, but can perform substantially faster? In the SLP-compressed setting this leads to the question of how to handle updates of SLP-compressed strings, which is an aspect mostly neglected in the classical literature on algorithmics on SLP-compressed strings. It has been shown in [38] that enumeration of regular spanners in the SLP-compressed setting extends well to the dynamic case.

Weighted Document Spanners: Another classical field of formal languages that also has a natural equivalent in document spanners is that of weighted automata. In the weighted automata setting, transitions are equipped with weights from an algebraic structure, which then extends automata from decision procedures to algorithms that produce quantitative outputs.

In the context of information extraction, representing spanners by weighted automata allows to allocate weights to the single span tuples of the span relation that is extracted by the spanner from an input word. In a database context, this is quite helpful, since it can be assumed that not all of the (potentially exponentially many) span tuples are equally relevant. Instead, it is likely that some span tuples are more interesting than others and it is desirable if the more important tuples appear first in an enumeration. In fact, an important motivation of the whole enumeration point of view is that the enumeration starts rather fast and can then be stopped by the user as soon as enough relevant outputs have been received. This point of view makes even more sense if the outputs of the enumeration are guaranteed to be produced in decreasing order with respect to their importance.

Such weighted settings of regular spanners and respective enumeration algorithms have been investigated in [12, 10, 8].

Other Areas in Database Theory Connected to Formal Languages: Despite the fact that many papers about document spanners have been published over the last decade (mostly in conferences ICDT and PODS), information extraction is still an active research area within the field of database theory.

Regular expressions (and therefore finite automata) are also a central tool for query classes for graph databases (see, e. g., [7, 5, 39]).

Typical approaches in the field of complex event processing (see [24, 26] for surveys) are often automaton-based (see, e. g., [23, 22, 21]).

3 Conclusions

Information extraction in database theory is a good example of how classical results of formal language theory can play a central role in new and emerging research topics. In particular, one can observe that document spanners is not just an application of established results, but rather poses new research questions about formal languages that are not covered by the existing literature.

References

[1] Margareta Ackerman and Jeffrey Shallit. Efficient enumeration of words in regular languages. *Theoretical Computer Science*, 410(37):3461–3470, 2009.

[2] Antoine Amarilli, Pierre Bourhis, Stefan Mengel, and Matthias Niewerth. Constant-delay enumeration for nondeterministic document spanners. *SIGMOD Rec.*, 49(1):25–32, 2020.

[3] Antoine Amarilli, Pierre Bourhis, Stefan Mengel, and Matthias Niewerth. Constant-delay enumeration for nondeterministic document spanners. *ACM Trans. Database Syst.*, 46(1):2:1–2:30, 2021.

[4] Antoine Amarilli, Louis Jachiet, Martin Muñoz, and Cristian Riveros. Efficient enumeration for annotated grammars. In *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 291–300, 2022.

[5] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Alastair Green, Jan Hidders, Bei Li, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Stefan Plantikow, Ognjen Savkovic, Michael Schmidt, Juan Sequeda, Slawek Staworko, Dominik Tomaszuk, Hannes Voigt, Domagoj Vrgoc, Mingxi Wu, and Dusan Zivkovic. Pg-schema: Schemas for property graphs. *Proc. ACM Manag. Data*, 1(2):198:1–198:25, 2023.

[6] Dana Angluin. Finding patterns common to a set of strings. *J. Comput. Syst. Sci.*, 21(1):46–62, 1980.

[7] Pablo Barceló. Querying graph databases. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, pages 175–188, 2013.

[8] Pierre Bourhis, Alejandro Grez, Louis Jachiet, and Cristian Riveros. Ranked enumeration of MSO logic on words. In *24th International Conference on Database Theory, ICDT 2021, March 23-26, 2021, Nicosia, Cyprus*, pages 20:1–20:19, 2021.

[9] Cezar Câmpeanu, Kai Salomaa, and Sheng Yu. Regex and extended regex. In *Implementation and Application of Automata, 7th International Conference, CIAA 2002, Tours, France, July 3-5, 2002, Revised Papers*, pages 77–84, 2002.

[10] Johannes Doleschal, Benny Kimelfeld, and Wim Martens. The complexity of aggregates over extractions by regular expressions. *Log. Methods Comput. Sci.*, 19(3), 2023.

[11] Johannes Doleschal, Benny Kimelfeld, Wim Martens, Yoav Nahshon, and Frank Neven. Split-correctness in information extraction. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 149–163, 2019.

[12] Johannes Doleschal, Benny Kimelfeld, Wim Martens, and Liat Peterfreund. Weight annotation in information extraction. In *23rd International Conference on Database Theory, ICDT 2020, March 30-April 2, 2020, Copenhagen, Denmark*, pages 8:1–8:18, 2020.

[13] R. Fagin, B. Kimelfeld, F. Reiss, and S. Vansummeren. Document spanners: A formal approach to information extraction. *J. ACM*, 62(2):12:1–12:51, 2015.

[14] Fernando Florenzano, Cristian Riveros, Martín Ugarte, Stijn Vansummeren, and Domagoj Vrgoc. Efficient enumeration algorithms for regular document spanners. *ACM Trans. Database Syst.*, 45(1):3:1–3:42, 2020.

[15] D. Freydenberger. A logic for document spanners. *Theory Comput. Syst.*, 63(7):1679–1754, 2019.

[16] D. Freydenberger and M. Holldack. Document spanners: From expressive power to decision problems. *Theory Comput. Syst.*, 62(4):854–898, 2018.

[17] Dominik D. Freydenberger. Extended regular expressions: Succinctness and decidability. *Theory of Computing Systems (ToCS)*, 53(2):159–193, 2013.

[18] Dominik D. Freydenberger, Benny Kimelfeld, and Liat Peterfreund. Joining extractions of regular expressions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 137–149, 2018.

[19] Dominik D. Freydenberger and Sam M. Thompson. Dynamic complexity of document spanners. In *23rd International Conference on Database Theory, ICDT 2020, March 30-April 2, 2020, Copenhagen, Denmark*, pages 11:1–11:21, 2020.

[20] Dominik D. Freydenberger and Sam M. Thompson. Splitting spanner atoms: A tool for acyclic core spanners. In *25th International Conference on Database Theory, ICDT 2022, March 29 to April 1, 2022, Edinburgh, UK (Virtual Conference)*, pages 10:1–10:18, 2022.

[21] Alejandro Grez and Cristian Riveros. Towards streaming evaluation of queries with correlation in complex event processing. In *23rd International Conference on Database Theory, ICDT 2020, March 30-April 2, 2020, Copenhagen, Denmark*, pages 14:1–14:17, 2020.

[22] Alejandro Grez, Cristian Riveros, Martín Ugarte, and Stijn Vansummeren. On the expressiveness of languages for complex event recognition. In *23rd International Conference on Database Theory, ICDT 2020, March 30-April 2, 2020, Copenhagen, Denmark*, pages 15:1–15:17, 2020.

[23] Alejandro Grez, Cristian Riveros, Martín Ugarte, and Stijn Vansummeren. A formal framework for complex event recognition. *ACM Trans. Database Syst.*, 46(4):16:1–16:49, 2021.

[24] Martin Hirzel, Guillaume Baudart, Angela Bonifati, Emanuele Della Valle, Sherif Sakr, and Akrivi Vlachou. Stream processing languages in the big data era. *SIGMOD Rec.*, 47(2):29–40, 2018.

[25] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation, 3rd Edition*. Pearson international edition. Addison-Wesley, 2007.

[26] Alessandro Margara and Gianpaolo Cugola. Processing flows of information: from data stream to complex event processing. In *Proceedings of the Fifth ACM International Conference on Distributed Event-Based Systems, DEBS 2011, New York, NY, USA, July 11-15, 2011*, pages 359–360, 2011.

[27] Francisco Maturana, Cristian Riveros, and Domagoj Vrgoc. Document spanners for extracting incomplete information: Expressiveness and complexity. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 125–136, 2018.

[28] Martin Muñoz and Cristian Riveros. Constant-delay enumeration for slp-compressed documents. In *26th International Conference on Database Theory, ICDT 2023, March 28-31, 2023, Ioannina, Greece*, pages 7:1–7:17, 2023.

[29] L. Peterfreund. *The Complexity of Relational Queries over Extractions from Text*. PhD thesis, 2019.

[30] Liat Peterfreund. Grammars for document spanners. In *24th International Conference on Database Theory, ICDT 2021, March 23-26, 2021, Nicosia, Cyprus*, pages 7:1–7:18, 2021.

[31] Liat Peterfreund, Dominik D. Freydenberger, Benny Kimelfeld, and Markus Kröll. Complexity bounds for relational algebra over document spanners. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019.*, pages 320–334, 2019.

[32] Liat Peterfreund, Balder ten Cate, Ronald Fagin, and Benny Kimelfeld. Recursive programs for document spanners. In *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*, pages 13:1–13:18, 2019.

[33] Markus L. Schmid. Characterising REGEX languages by regular languages equipped with factor-referencing. *Information and Computation (I&C)*, 249:1–17, 2016.

[34] Markus L. Schmid. The information extraction framework of document spanners - A very informal survey. In *SOFSEM 2024: Theory and Practice of Computer Science - 49th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2024, Cochem, Germany, February 19-23, 2024, Proceedings*, pages 3–22, 2024.

[35] Markus L. Schmid and Nicole Schweikardt. A purely regular approach to non-regular core spanners. In *24th International Conference on Database Theory, ICDT 2021, March 23-26, 2021, Nicosia, Cyprus*, pages 4:1–4:19, 2021.

[36] Markus L. Schmid and Nicole Schweikardt. Spanner evaluation over slp-compressed documents. In *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, pages 153–165, 2021.

[37] Markus L. Schmid and Nicole Schweikardt. Document spanners - A brief overview of concepts, results, and recent developments. In *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 139–150, 2022.

[38] Markus L. Schmid and Nicole Schweikardt. Query evaluation over slp-represented document databases with complex document editing. In *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 79–89, 2022.

[39] Domagoj Vrgoc, Carlos Rojas, Renzo Angles, Marcelo Arenas, Diego Arroyuelo, Carlos Buil-Aranda, Aidan Hogan, Gonzalo Navarro, Cristian Riveros, and Juan Romero. Millenniumdb: An open-source graph database system. *Data Intell.*, 5(3):560–610, 2023.

The Logic in Computer Science Column

by

Yuri Gurevich

Computer Science & Engineering
University of Michigan, Ann Arbor, Michigan, USA
gurevich@umich.edu

Foreword by the columnist

The ongoing AI revolution raises many foundational problems. For quite a while, I felt that the issue needs to be addressed in this column. Not being an AI expert, I was looking for volunteers. This didn't work, and so one day I took a deep breath and started to write an article myself. Andreas Blass, my long-time collaborator, was reluctant to join me, but eventually he agreed.

A hundred years ago, logic was almost synonymous with foundational studies. I tried to rekindle that tradition in [5]. The goal of the following dialog is to provoke young logicians with a taste for foundations to notice the foundational problems raised by the ongoing AI revolution.

A whimsical picture of a neural net with a devil's head by the OpenAI tool Dall-E

On logic and generative AI

Yuri Gurevich and Andreas Blass
University of Michigan in Ann Arbor, MI, USA

I think the most beautiful thing about
deep learning is that it actually works.
—Ilya Sutskever [13, 29:46]

§1 Thinking fast and slow[1]

Q: I just learned that Daniel Kahneman, Nobel laureate in economics and the author of "Thinking, fast and slow" [7], passed away on March 27, 2024. I heard a lot about this book but have never read it. What did he mean by thinking fast and thinking slow?

A: Daniel Kahneman and Amos Tversky discovered that human thinking is driven by two distinct systems, System 1 and System 2.

System 1 supports *fast thinking*. It "operates automatically and quickly, with little or no effort and no sense of voluntary control ... The capabilities of System 1 include innate skills that we share with other animals" [7, pp. 41-43]. System 1 is good at detecting patterns and reading situations on the fly. It allows us to make snap judgments and decisions without deliberation, but it is prone to biases and errors.

System 2 supports *slow thinking* which is deliberate, analytical, and conscious. "System 2 allocates attention to the effortful mental activities that demand it, including complex computations. The operations of System 2 are often associated with the subjective experience of agency, choice, and concentration" [7, p 41]. System 2 can override the impulses and biases generated by System 1.

Q: I probably rely on System 1 more than I should.

A: System 2 is much slower than System 1 and requires more effort. "While walking comfortably with a friend, ask him to compute 23×78 in his head, and to do so immediately. He will almost certainly stop in his tracks " [7, p 74]. It is not surprising that often we tend to be lazy and rely on System 1 more than we should.

Q: Have either of you met Kahneman?

[1]The freewheeling conversations [1, 4, 10] are broken into "chapters" for the reader's convenience. Following this example, our freewheeling dialog was broken into parts at the eleventh hour.
Below Q is Quisani, a former student of the first author, and A is the authors.

A: Andreas: No, I haven't.
Yuri: Neither have I, even though, in the 1970s, when I was teaching at Ben Gurion University, I had a good chance to see Kahneman and Tversky at the Hebrew University of Jerusalem where they had their seminar in psychology. I was at the Hebrew University on Wednesdays. My Jerusalem colleagues spoke enthusiastically about the psychology seminar, and I intended to drop in some day, but never did. Those Wednesdays were too busy for me: model theory seminar, set theory seminar, and work with the renowned logician Saharon Shelah.

§2 Is generative AI intelligent?

Q: Generative AI is getting a lot of press recently. Is it intelligent?

A: The AI revolution is impressive. Generative AI is used in many domains. In particular, it is used to improve existing logic tools. Generative AI is changing our lives. But is it already intelligent? The issue is controversial and super interesting; it richly deserves a separate conversation.

Q: May we address it anyway, at least briefly?

A: Sure. So the thesis under discussion is that the current generative AI is intelligent. Let's first hear skeptics.

Edward Gibson is a psycholinguistics professor at MIT and the head of the MIT Language Lab. In his recent interview with Lex Fridman, Gibson argues that the current LLMs are all about the form, not meaning.

> "I would argue they're doing the form. They're doing the form, they're doing it really, really well. And are they doing the meaning? No, probably not. There's lots of these examples from various groups showing that they can be tricked in all kinds of ways. They really don't understand the meaning of what's going on. And so there's a lot of examples … which show they don't really understand what's going on" [4, 01:34:33].

Q: Presumably, GPT-4 knows all grammatical rules of English in all existing grammar books.

A: The rules of natural languages may have statistical character and may not be written anywhere.

Q: Give me an example.

`A:` A typical American pronounces the one vowel in "Blass" as in "glass" and stresses the first syllable in "Gurevich." In fact, "Blass" is a German name, and German doesn't have the English "glass" vowel. Similarly, "Gurevich" is a Russian name, and Russians stress the second syllable.

`Q:` This is about pronunciation. Give me a text-based example.

`A:` One has **a** cold, **the** flu, or influenza.

`Q:` Wow, I am glad that I don't have to teach grammar.

`A:` Also, in America, one is "in the hospital" while, in England, one is "in hospital." In America, the government "proposes" a law, while, in England, the government "propose" a law.

`Q:` You mentioned "skeptics." Whom else did you have in mind?

`A:` Yann LeCun, along with Yoshua Bengio and Geoffrey Hinton, received the 2018 Turing Award for their work on deep learning. They are sometimes referred to as the "Godfathers of Deep Learning." LeCun says that LLMs, large language models like GPT-4, aren't truly intelligent yet (and cannot take us to superhuman intelligence) because they essentially lack important capabilities [10, 00:02:47]:

1. understanding the physical world,
2. persistent memory,
3. reasoning, and
4. planning.

"That is not to say," adds LeCun, "that autoregressive LLMs are not useful, they're certainly useful; or that they're not interesting; or that we can't build a whole ecosystem of applications around them, of course we can. But as a path towards human-level intelligence, they're missing essential components."

At this point, we heard two skeptics. What do you think?

`Q:` Gibson's argument did not convince me too much. I am thinking about clever extraterrestrials analysing the wet chemistry of our nervous system and wondering do humans really understand the meaning of what's going on? There are lots of ways humans can be tricked into irrational behaviour.

LeCun doesn't look to me like a true skeptic. He points out various ways to improve the current generative AI.

Also, there is a matter of definitions, in particular the definition of intelligence. Maybe we should be speaking about degrees of intelligence.

A: OK, let's turn our attention to the defence of the thesis. In a recent talk, Geoffrey Hinton made a strong case for the intelligence of the current large language models, LLMs.

> "They [LLMs] turn words into features, they make these features interact, and from those feature interactions they predict the features of the next word. And what I want to claim is that these millions of features and billions of interactions between features that they learn, are understanding ...
>
> This is the best model we have of how we understand. So it's not like there's this weird way of understanding that these AI systems are doing and then this [is] how the brain does it. The best that we have, of how the brain does it, is by assigning features to words and having features [and] interactions" [6, 0:14–15].

Q: But these allegedly intelligent LLMs hallucinate.

A: So do we. "Anybody who's studied memory, going back to Bartlett in the 1930s, knows that people are actually just like these large language models. They just invent stuff and for us, there's no hard line between a true memory and a false memory" [6, 0:16].

Q: Interesting. I intend to listen to that talk of Hinton.

A: Another champion of the thesis is Ilya Sutskever, Chief Scientist at Open AI [13, 14].

A brief discussion cannot do justice to an issue as complicated as the thesis. Let's move on.

Q: Wait, there is a closely related issue: What about the future? Many AI experts say that generative AI is getting more intelligent than humans and may become a danger to us.

A: The chances are that its intelligence will be incomparable to ours. Think of airplanes versus birds. Airplanes fly faster, but birds can land and take off almost anywhere. On the other hand, AI develops fast and we don't know the directions. "It's tough to make predictions, especially about the future," said the baseball philosopher Yogi Berra.

In any case, you are right. Numerous AI experts worry that generative AI may become a danger to humans. They point out that it may be used by bad actors for manipulating electorates and waging wars.

> “There are always misguided or ill-intentioned people,” says Bengio, “so it seems highly probable that at least one organisation or person would — intentionally or not — misuse a powerful tool once it became widely available” [2].

In the talk [6], Hinton describes various scenarios he is worried about.

> “But the threat I’m really worried about,” says Hinton, “is the long term existential threat. That is the threat that these things could wipe out humanity . . .
>
> [W]hat happens if superintelligences compete with each other? . . . The one that can grab the most resources will become the smartest. As soon as they get any sense of self-preservation, then you’ll get evolution occurring. The ones with more sense of self-preservation will win and the more aggressive ones will win” [6, 0:21–24].

`Q:` This is frightening indeed.

§3 Reasoning and logic in AI

`Q:` I am fascinated by LeCun’s analysis above. Let’s look at it more carefully. Persistent memory sounds technical, not as profound as the three other capabilities.

`A:` Persistent memory would help with long-term coherence. If you heavily interact with an LLM for a long time, it would be highly desirable that the LLM maintains coherence and consistency. This is a challenge for the current systems.

`Q:` This is a challenge for some humans as well ☺

`A:` Also, in humans, persistent memory is vital for incubation of ideas, cross-pollination of ideas from different domains, and reasoning by analogy.

`Q:` Concerning LeCun’s item 3, what kind of reasoning is AI incapable of?

`A:` System 2 reasoning. An LLM is basically just two files. One is a huge data file that reflects the information the model was trained on. The other is an algorithm, typically succinct, which usually embodies the model architecture as well as a probabilistic inference mechanism. That mechanism works with a sequence of words. It starts with a given query, and runs in rounds. During one round, it infers another word and appends it to the current sequence. (More exactly, it works with subword tokens which are not necessarily full words.)

Sometimes the next word is obvious, but sometimes it is very hard to figure out an appropriate next word. But the LLM cannot stop and think. In the AI parlance, "it is allocating approximately the same amount of compute for each token it generates" [1, 00:59:51]. In that sense it uses only fast thinking (System 1). As far as we know, it is an open problem how to improve the process by incorporating slow, deliberate thinking (System 2).

`Q:` Understanding of the real world seems to involve reasoning as well, both System 1 reasoning and System 2 reasoning, from evading predators (fast thinking) to quantum physics (slow thinking). Thus reasoning is ubiquitous in AI, and AI needs many different kinds of reasoning. It would be natural if the role of logic in AI would grow and grow. And yet, Yuri wrote that "the golden age of logic in artificial intelligence is behind us" [5, §4]. How come?

`A:` At the early stage of AI, the logic approach was dominating. A lot of good work was done; see the article "Logic-based artificial intelligence" in the Stanford Encyclopedia of Philosophy [15]. But then an important competitor arose. Let us quote LeCun on the issue[2].

> "In the 1950s, while the heralds of classical AI, based on logic and tree-based exploration, were pushing back its limits, the pioneers of learning started to make their voices heard. They defended the idea that, if we want to make computer systems capable, like animals and humans, of complex tasks, logic is not enough. We must get closer to the functioning of the brain, and therefore make the systems capable of self-programming, drawing inspiration from their learning mechanisms. This part of research based on deep learning and (artificial) neural networks is the one to which I am devoted here. It is at work in all of today's spectacular applications, starting with the autonomous car" [9, p. 23].

[2]The French original: "Dans les années 1950, tandis que les héraults de l'intelligence artificielle classique, basée sur la logique et l'exploration arborescente, en repoussent les limites, les pionniers de l'apprentissage commencent à donner de la voix. Ils défendent l'idée que, si l'on veut rendre les systèmes informatiques capables de tâches complexes, à la manière de l'animal et de l'homme, la logique ne suffit pas. Il faut se rapprocher du fonctionnement du cerveau, et donc rendre les systèmes capables de se programmer eux-mêmes, en s'inspirant de ses mécanismes d'apprentissage. Cette partie de la recherche fondée sur l'apprentissage profond (deep learning) et les réseaux de neurones (artificiels) est celle à laquelle je me consacre. Elle est à l'œuvre dans toutes les applications spectaculaires actuelles, à commencer par la voiture autonome" [9, p. 23].

§4 Real-world understanding

`Q:` Coming back to real-world understanding, robots need to navigate around obstacles and handle objects without causing damage to those objects, to themselves, and of course to people. This requires understanding of the physical world. But, unlike robots, LLMs don't walk around and they have read much of what there is to read. Shouldn't all that knowledge give them a near perfect understanding of the physical world?

`A:` Not really.

`Q:` Can you illustrate that?

`A:` Sure. Henri Lebesgue had an idea that we can build upon: there are cases where arithmetic does not apply. "Suppose I put two animals in a cage and then two more. How many animals will the cage then contain? [8, p. 14]" Let's present a version of this scenario to Google's Gemini.

— Suppose that I put 2 animals of one kind and 3 animals of another kind into a closed space. How many animals will be there altogether?

— Adding the number of animals of each kind together, you will have 2 + 3 = 5 animals altogether.

`Q:` This is ridiculous. Gemini doesn't realize that the result depends on the kinds of animals. For example, the two animals of the first kind may consume some or all of the three animals of the other kind. Maybe, Gemini takes into account only the moment that the you close the door of the closed space.

`A:` OK, let's check your hypothesis.

— What if I check the total number of animals not immediately but in one hour, one day, or one month?

— The total number of animals (5) wouldn't change regardless of whether you check on them in one hour, one day, or even one month. As long as no animals enter or leave the closed space, the initial sum of 2 and 3 will remain the same.

`Q:` One month is not a trivial period. Even if none of our original animals is a predator, some of the animals may die or have young.

A: Gemini doesn't have your real-world experience[3].

Q: Have you seen imperfections with other LLMs?

A: In §3, we mentioned an article on logic-based AI in the Stanford Encyclopedia of Philosophy (SEP). Before turning to SEP, we asked GPT-4 to suggest a good survey article on logic-based AI. GPT-4 hallucinated a combination of an existing title and well-known authors who could write an article with this title but didn't.

Q: This harks back to the issue that an LLM can't stop and think.

A: In this particular case, the necessary thinking is algorithmic: just check the known sources. So this hallucination seems to be a bug.

§5 Moravec's paradox

Q: It is ironic that Gemini can do impressive intellectual work but misses obvious things about the physical world.

A: You rediscovered Moravec's paradox from the 1980s. Here's an instructive quote from Hans Moravec's 1988 book [11, pp. 15–16]:

> "It seemed to me that, in the early 1970s, some of the creators of successful reasoning programs suspected that the poor performance in the robotics work somehow reflected the intellectual abilities of those attracted to that side of the research. Such intellectual snobbery is not unheard of, for instance between theorists and experimentalists in physics. But as the number of demonstrations has mounted, it has become clear that it is comparatively easy to make computers exhibit adult-level performance in solving problems on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility.
>
> In hindsight, this dichotomy is not surprising. Since the first multicelled animals appeared about a billion years ago, survival in the fierce competition over such limited resources as space, food, or mates has often been awarded to the animal that could most quickly produce a correct action from inconclusive perceptions. Encoded in the large, highly evolved sensory and motor portions of the human

[3]The chat with Gemini occurred on April 29th. Long before that, we presented to Gemini another scenario (a coin tossed down from a tall building) that also exposed Gemini's lack of real-world knowledge. In early May, we again presented both scenarios to Gemini and discovered that Gemini improved in the meantime. Despite the improvements, Gemini still showed, albeit in a less dramatic way, a lack of real-world knowledge.

> brain is a billion years of experience about the nature of the world and how to survive in it. The deliberate process we call reasoning is, I believe, the thinnest veneer of human thought, effective only because it is supported by this much older and much more powerful, though usually unconscious, sensorimotor knowledge. We are all prodigious olympians in perceptual and motor areas, so good that we make the difficult look easy. Abstract thought, though, is a new trick, perhaps less than 100 thousand years old. We have not yet mastered it. It is not all that intrinsically difficult; it just seems so when we do it."

`Q:` I don't buy that slow thinking is not all that intrinsically difficult. He also seems to suggest that — given a chance and time — evolution will make slow thinking more efficient. Taking into account how wayward evolution is, the time in question may be humongous. But I digress.

`A:` In the Lex Fridman podcast, LeCun further develops Moravec's argument.

> "Those LLMs are trained on enormous amounts of texts, basically, the entirety of all publicly available texts on the internet, right? That's typically on the order of 10^{13} tokens. Each token is typically two bytes, so that's $2 \cdot 10^{13}$ bytes as training data. It would take you or me 170,000 years to just read through this at eight hours a day. So it seems like an enormous amount of knowledge that those systems can accumulate, but then you realize it's really not that much data. If you talk to developmental psychologists, they tell you [that] a four-year-old has been awake for 16,000 hours in his or her life, and the amount of information that has reached the visual cortex of that child in four years is about 10^{15} bytes [10, 00:04:08] . . .
>
> What that tells you is that through sensory input we see a lot more information than we do through language and that, despite our intuition, most of what we learn and most of our knowledge is through our observation and interaction with the real world, not through language" [10, 00:05:12].

§6 Thinking fast

`Q:` All this makes the fast-thinking grasp of the real world even more interesting and relevant. Clearly, LLMs would need such a grasp.

`A:` They need it already. Have you recently spoken to an LLM-powered chatbot?

`Q:` I try to avoid them and talk to humans.

`A:` This will be harder and harder to do.

Q: It is already plenty hard and sometimes virtually impossible; try to talk to OpenAI support.

But, OK, let's suppose that I am chatting with a bot representative of some company.

A: Imagine that they sent you something, say a laptop, and it arrived damaged, that you waited for that laptop longer than promised, and that you need one right now. A human would quickly realize that you are upset and would try to be extra empathetic. So should the bot.

Q: Yes, and this requires reading the situation on the fly, fast thinking.

Now that I have some idea about fast thinking, I understand Yuri's question what the logic of fast thinking is in the February 2021 column. But is there is a logic of fast thinking?

A: Much depends on what we mean by logic. George Boole discovered an algebra of universal "laws of thought" [3], that is, universal laws of slow thinking. Is there a useful algebra of universal laws of fast thinking? We doubt it. Fast thinking is rather far from exact reasoning.

Q: Are there any meaningful rules of fast thinking?

A: Oh, yes. For example, we intuitively judge the frequency of an event by how easily examples come to mind, which can be influenced by recent exposure and emotions [7, p. 244]. Such rules of thumb help us, given inconclusive perceptions, to produce quickly an action that is often beneficial but may be detrimental.

Arguably logic should study the laws of thinking, including fast thinking, whatever they are. The intention, in the question about the logic of fast thinking, was to provoke young logicians with a taste for foundations to notice the nascent study of fast thinking.

Q: Studying fast thinking may expand logic tremendously. Do you know historical examples where logic expanded so greatly?

§7 The dawn and heyday of logic

A: One example is related to the emergence of logic as a separate discipline in the classical Greek period.

Q: Was there any logic before that?

A: Various forms of structured argumentation were used before the classical Greek period, certainly in China and India. But it was in Greece that logic became a separate discipline.

Q: Does this have something to do with Athenian democracy?

A: The practice of Greek democracy wasn't limited to Athens, though Athens is the best-documented case. The Athenian democracy was characterized by the participation of its (free male) citizens in the Assembly and courts. There were many occasions where one had to convince numerous citizens rather than talking to gods or to a few aristocrats. The necessity of convincing argumentation drove the development of logic as well as of rhetoric and demagoguery. The contributions of Aristotle and his school were particularly significant in the birth of logic as a discipline. Aristotle's system of logic was virtually unchallenged until the 19th century.

Q: Mathematics greatly advanced in the period from Aristotle to the 19th century. Mathematical arguments went beyond Aristotelian syllogisms. They must have used much more sophisticated logic, and logic seems to precede mathematics, logically speaking ☺

A: Yes, already in the time of Aristotle mathematics involved logic beyond Aristotelian syllogisms. But for a long time mathematicians used deduction principles implicitly. After all, those principles are essentially trivialities. One learned them in the course of learning mathematics.

Q: Still, there were many developments in mathematics, between Euclid's time and 19th century.

A: The most dramatic development was the invention of calculus. Its use of infinitesimals and related notions, like infinite sums, certainly raised new logical issues. Reasonable looking computations could contradict each other.

Q: Give me an example.

A: Consider the infinite series $1 - 1 + 1 - 1 + 1 - 1 + \dots$ One way to sum it is $(1-1) + (1-1) + (1-1) + \cdots = 0$. Another way is $1 + (-1+1) + (-1+1) + \cdots = 1$. Which way, if any, is the correct one? The foundations of mathematics became problematic.

Q: Today we know that the series does not converge.

A: It took a while for mathematicians to work out notions like convergence. The problem was actual infinity. Mathematicians used to work with finite objects and potential infinity, never actual infinity. E.g., for Euclid, the only lines were finite line segments.

Q: In my experience, people mention infinitesimals but don't use them.

A: Eventually, the language of infinitesimals was replaced by the precise language of epsilon and delta, and thus the foundations of analysis got rid of the fuzziness induced by infinitesimals.

The introduction of set theory by Georg Cantor improved the situation further. Cantor bravely dealt with actual infinity and made great progress. Mathematicians quickly grew comfortable with set theory. In particular, it was observed that ordered pairs can be coded as sets, which allowed modern set-theoretic definitions of notions like *function* and *relation* purely in terms of set membership.

But at the same time paradoxes of set theory were discovered, and a real crisis arose in the foundations of mathematics. It wasn't clear anymore which convincing reasoning is valid and which is not.

Q: And, I guess, all this provoked the birth of mathematical logic.

A: Even before the discovery of paradoxes, important foundational work was done, notably by George Boole, Giuseppe Peano, and Gottlob Frege. But the foundational crisis put mathematical logic in the center of attention. The crisis was resolved, for all practical purposes, by the development of axiomatic set theory.

Q: Did this make logic popular?

A: It surely did. Andrei Kolmogorov and John von Neumann started as logicians. Alan Turing wrote his PhD thesis in logic [16]. The universal Turing machine suggested to von Neumann the idea of the universal electronic computer which made logic even more popular.

In 1956, when the famous Dartmouth workshop [17] started AI as a field, logic was at the zenith of its popularity. This is a partial explanation for the dominance of the logic approach at the early stage of AI. By the way, the term "artificial intelligence" was coined by John McCarthy, Marvin Minsky, Nathaniel Rochester and Claude E. Shannon in their proposal for the Dartmouth conference.

§8 Logic in the age of AI

Q: In both cases, the birth of logic and the resolution of the foundational crisis in mathematics, there was a need, and logic rose to the occasion. Demand preceded supply, so to speak. Do you think that, this time around, generative AI provides sufficient demand?

A: It does. The demand is ubiquitous.

Q: Give me an example.

A: A quick glance at a person gives you a surprising amount of information. We may learn the gender, estimate the age, etc. Such rapid assessment was instrumental to our survival as a species.

Q: It seems that, as in object-oriented programming, a person *P* is viewed as an object with various attributes.

A: But the set of attributes may depend on the assessor; for example, a runner may see whether *P* is a runner. Also, the set of values that an attribute takes may be dynamic; for example, a child may learn only whether *P* is a child or an adult — in contrast to an adult's more precise estimate of *P*'s age.

Q: Surely, psychologists and neuroscientists are already studying rapid assessment. What can logic add?

A: Rapid assessment is a case of fast thinking. In [5], Yuri asked what the logic of fast thinking is. If and when laws of fast thinking are understood, at least to some extent, they may help us to analyze rapid assessment.

Q: Do you think that logic will rise to meet the AI challenge?

A: This is an interesting question. To us, logic is the study of reasoning, any kind of reasoning [5]. There is no doubt that the kinds of reasoning which are relevant to AI will be studied.

Q: But will those studies be called logic? Maybe, AI logic?

A: At the moment this does not look probable. It is not predetermined what is or is not logic. Conflicting social processes are in play. Much depends on how — and whether — the logic community will address the challenge.

The story of infinitesimal calculus may be instructive. See a sketch of that story in [5, §2]. After the invention of infinitesimal calculus, a question arose how to reason with infinitesimals? For a couple of centuries, mathematicians struggled with the problem and eventually largely solved it using the epsilon-delta approach that essentially eliminated infinitesimals. In a sense, the mathematical heroes of that story worked as logicians but this wasn't recognized. This particular story has a happy, albeit somewhat bitter, end of sorts as far as logicians are concerned. Abraham Robinson came up with a proper logic of infinitesimals, which he called nonstandard analysis and which allows you to handle infinitesimals consistently [12]. But, in the meantime, mathematicians got accustomed to the epsilon-delta approach and weren't too much interested in nonstandard analysis which is arguably more elegant and efficient.

Q: You wish of course that logic will be more successful in AI.

`A:` True.

Acknowledgments

Many thanks to Naomi Gurevich, Ben Kuipers, Vladimir Lifschitz, and Moshe Vardi for useful comments.

References

[1] Sam Altman, "OpenAI, GPT-5, Sora, Board Saga, Elon Musk, Ilya, Power, and AGI," *Lex Fridman Podcast* #419, 18 March 2024, transcript, `https://lexfridman.com/sam-altman-2`

[2] Yoshua Bengio, "One of the 'godfathers of AI' airs his concerns," *The Economist* July 21st 2023

[3] George Boole, "An investigation of the laws of thought," Walton & Maberly 1854, `https://gutenberg.org/ebooks/15114`

[4] Edward Gibson "Human language, psycholinguistics, syntax, grammar, & LLMs," *Lex Fridman Podcast* #426, 17 April 2024, transcript, `https://lexfridman.com/edward-gibson-transcript`
(At the time of writing, the transcript mistakenly assigned Gibson's words to Fridman and vice versa.)

[5] Yuri Gurevich, "Logical foundations: Personal perspective," *Bulletin of EATCS* February 2021, also arXiv:2103.03930 and *Logic Journal of the IGPL* 33:6 Dec 2023 1192–1202

[6] Geoffrey Hinton, "Will digital intelligence replace biological intelligence," Romanes Lecture, Oxford, UK, 19 February 2024
`https://www.youtube.com/watch?v=N1TEjTeQeg0&list=PPSV`

[7] Daniel Kahneman, "Thinking fast and slow," Farrar, Straus and Giroux 2011

[8] Henri Lebesgue, "Measure and the integral" (a translation of "La mesure des grandeurs" and "Sur le développement de la théorie de l'intégrale"), Holden-Day 1966

[9] Yann LeCun, "Quand la machine apprend," avec la collaboration de Caroline Brizard, Odile Jacob 2019. Russian translation: Ян Лекун, "Как учится машина," ПРО Москва 2021. There is no English translation yet, it seems.

[10] Yann LeCun, "Meta AI, Open Source, Limits of LLMs, AGI & the Future of AI," *Lex Fridman Podcast* #416, 7 March 2024, transcript, `https://lexfridman.com/yann-lecun-3-transcript`

[11] Hans Moravec, "Mind children," Harvard University Press 1988

[12] Abraham Robinson, "Non-standard analysis," North-Holland 1966

[13] Ilya Sutskever, “Deep learning,” *Lex Fridman Podcast* #94, 8 May 2020, `https://www.youtube.com/watch?v=13CZPWmke6A` (At the time of writing, only an auto-generated transcript is available.)

[14] Ilya Sutskever, “The Exciting, Perilous Journey Toward AGI,” *TED Podcast* 20 November 2023,
`https://www.youtube.com/watch?v=SEkGLj0bwAU`

[15] Richmond Thomason, “Logic-based artificial intelligence,” *Stanford Encyclopedia of Philosophy* (Spring 2024 Edition), E.N. Zalta & U. Nodelman (eds.),
`https://plato.stanford.edu/archives/spr2024/entries/logic-ai/`

[16] Alan Turing, “Systems of logic based on ordinals,”
PhD Thesis, Princeton University 1938,
Proceedings of the London Mathematcal Society, Series 2 45, 161–228, 1939

[17] Wikipedia contributors, “Dartmouth workshop,”
`https://en.wikipedia.org/wiki/Dartmouth_workshop`.
The version, last edited on 12 April 2024, was retrieved May 10 2024.

The Distributed Computing Column

Seth Gilbert
National University of Singapore
seth.gilbert@comp.nus.edu.sg

This month, in the Distributed Computing Column, Helen Xu is discussing everything you need to know about how to design efficient parallel applications that operate on dynamic graphs! She focuses on three key aspects: (1) How do you design the *containers* (i.e., data structures) that encapsulate the dynamic graph? (2) What is the right framework (i.e., interface) for interacting with a dynamic graph? (3) How do you fairly benchmark the performance of your parallel application for dynamic graphs?

To answer these questions, she discusses two main results. First, she presents a new container for dynamic graphs called *F-Graph*. F-Graph is a multicore batch-parallel dynamic-graph system that is optimized for spatial locality. It is built on top of a batch-parallel packed-memory array, yielding fast performance for a variety of graph applications. Next, she presents BYO, a unified framework for large-scale graph containers designed to facilitate benchmarking. BYO provides a simple and abstract container API, along with a clean interface. She then uses BYO to evaluate 27 different graph containers on 10 different graph algorithms using 10 large graph datasets. The resulting data illuminates the issues and tradeoffs involved in designing parallel applications for dynamic graphs.

Overall, then, this article by Helen Xu provides significant insight into both the theory and practice of efficient parallel computation for dynamic graphs.

The Distributed Computing Column is particularly interested in contributions that propose interesting new directions and summarize important open problems in areas of interest. If you would like to write such a column, please contact me.

Dynamic Graphs, End-to-End: Containers, Frameworks, and Benchmarks

Helen Xu
Georgia Institute of Technology

1 Introduction to dynamic-graph containers

Dynamic graphs, or graphs that change over time, underlie many applications such as social networks [7], protein interactions [6], and paper citation networks [45]. Systems for dynamic graphs need to efficiently 1) apply a stream of updates (e.g., edge insertions and deletions) and 2) run queries (i.e., algorithms) on the updated graph. Unfortunately, these two objectives often conflict with each other [86].

In this column, I will address two main questions: 1) how to develop efficient parallel dynamic-graph data structures, or ***containers***, that support both fast updates and fast queries and 2) how to comprehensively and fairly benchmark dynamic-graph containers.

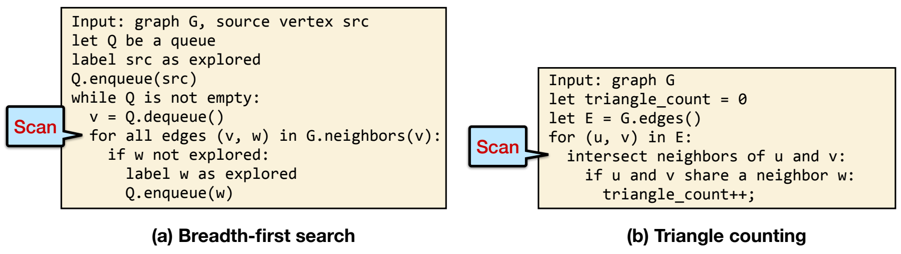
Query-update tradeoff. Let us examine several canonical graph containers to concretely understand the tradeoff between updates and queries. These examples will illustrate the challenges to supporting both fast updates and fast queries without giving up performance along either axis.

Perhaps the most classical graph data structure is the ***adjacency matrix***. Given a graph with n vertices, the associated adjacency matrix is an $n \times n$ matrix A where element A_{uv} is 1 if there is an edge from vertex u to vertex v, and 0 otherwise. The adjacency matrix format is ideal for edge updates - adding an edge to a graph stored in this format takes only $O(1)$ time to find and set the correct location in the matrix. Finding the existence of any particular edge also takes $O(1)$ time. It would seem that if one had $O(n^2)$ space, the adjacency matrix would be an ideal graph container.

In practice, however, the main operation underlying graph algorithms is a ***scan***, or iteration, through a vertex's neighbors, rather than individual edge-existence queries [69]. One common pattern in graph algorithms is processing a source vertex and adding its neighbors to the "active set" for the next round of processing. Finding and adding all neighbors of a given vertex to a set requires a vertex scan. Figure 1 provides concrete examples of how vertex scans are the main workhorse of graph algorithms.

Furthermore, many real-world graphs exhibit ***sparsity***: i.e., they contain many fewer edges than the total possible number of edges [64]. That is, almost all vertices have degree much less than $O(n)$. For example, the LiveJournal graph [7] has about 4.8 million vertices, but its average degree is only about 18. Therefore, an ideal graph data structure would support a scan of a given vertex v's neighbors in $O(\text{degree}(v))$ time. However, an adjacency matrix requires $O(n)$ time to scan the neighbors of any vertex, regardless of degree.

Therefore, the most popular graph data structure in high-performance graph applications is not the adjacency matrix but rather the ***Compressed Sparse Row*** (CSR) [72] representation. CSR stores a graph with m edges and n vertices in two arrays: an edge array A to store m neighbor


Scan
Scan
(a) Breadth-first search
(b) Triangle counting

```
Input: graph G, source vertex src
let Q be a queue
label src as explored
Q.enqueue(src)
while Q is not empty:
  v = Q.dequeue()
  for all edges (v, w) in G.neighbors(v):
    if w not explored:
      label w as explored
      Q.enqueue(w)
```

```
Input: graph G
let triangle_count = 0
let E = G.edges()
for (u, v) in E:
  intersect neighbors of u and v:
    if u and v share a neighbor w:
      triangle_count++;
```

Figure 1: Example of how vertex scans underlie two fundamental graph algorithms: (a) breadth-first search and (b) triangle counting.

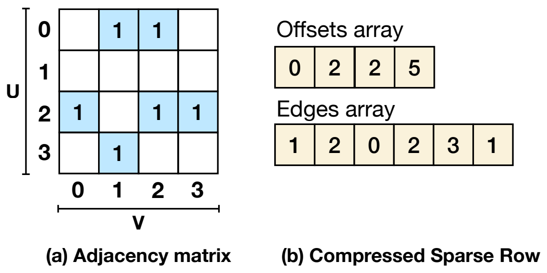


Figure 2: Example of the same graph stored in (a) adjacency matrix format and (b) Compressed Sparse Row format.

ids (for the edges) and an offsets array O to store n start indices (one per vertex). The neighbors of each vertex v form a contiguous fragment of A, so scanning over a given vertex v's neighbors only requires $O(\text{degree}(v))$ time. Furthermore, the neighbor ids of the edges are laid out contiguously in memory, so CSR exhibits good spatial locality and supports fast scans. However, CSR was not designed for dynamic graphs and is therefore prohibitively expensive to update: inserting a single edge may require $\Theta(m)$ time to move all of the other edges in the contiguous edge array. Figure 2 provides an example of a graph stored in adjacency matrix and CSR format.

High-performance dynamic-graph containers for multicores. To address this limitation, significant research effort over the past decade has been devoted to developing efficient dynamic-graph containers [38, 53, 32, 54, 34, 63, 75, 78, 82, 83, 33, 80, 52, 77, 43, 67, 24, 66, 84, 87].

The goal of this line of work is to introduce data structures to support both **efficient updates and algorithms** for dynamic graphs. Figure 3 provides a cartoon illustration of the query-update tradeoff and intuition for the data-structure design objectives.

When it comes to developing fast dynamic-graph data structures on modern multicores, codes must be optimized for the core features of modern multicore machines - many threads, large main memories, and a steep memory hierarchy [86]. Since today's multicores include large main memories (possibly 1TB or more), many of the dynamic-graph containers reside in memory. As we shall see, in-memory dynamic-graph containers must be optimized for locality to efficiently make use of both the memory subsystem as well as parallelism in multicores.

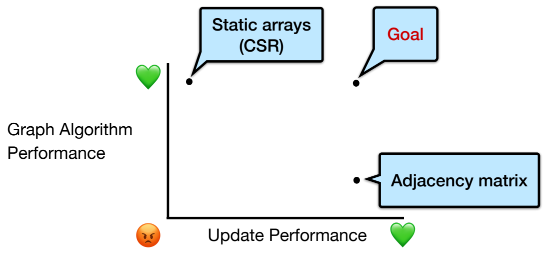


Figure 3: The query-update tradeoff and the goal of dynamic-graph containers.

2 F-Graph: A dynamic-graph system based on the Compressed Packed Memory Array

This section describes F-Graph, a multicore batch-parallel dynamic-graph system, as a case study for how to create efficient parallel dynamic-graph containers by optimizing for spatial locality [80][1]. As we will see, F-Graph overcomes the query-update tradeoff with a cache-optimized dynamic-array data structure. It supports both faster algorithms and faster updates compared to state-of-the-art dynamic-tree-based graph containers.

The associated artifact[2] was honored with the Best Artifact Award at PPoPP '24.

Batch-parallel dynamic-graph containers

Many modern data structure libraries, including dynamic-graph containers, parallelize ***batch updates*** that insert or delete multiple elements rather than point updates because each individual update is usually not worth parallelization due to their sublinear complexity [42, 9, 33, 34, 71, 39, 74, 8, 55]. Direct algorithmic support for batch updates simplifies parallelism and reduces the work per update by amortizing shared work between updates (e.g., searches for the target location in the data structure).

Batch-parallel data structures demonstrate the important role that the memory subsystem plays in good utilization of multithreaded parallelism and overall performance in practice. Almost all batch-parallel data structure implementations are based on pointer-based data structures such as trees [33, 34, 17, 42, 9, 71, 39, 74]. However, the main bottleneck in the parallel scalability of these implementations is not parallelism but memory bandwidth limitations due to pointer indirections during tree traversal [17, 42]. Dhulipala *et al.* [34, 33] address these issues by adding blocking and compression to improve spatial locality in trees. These batch-parallel cache-optimized trees form the basis for Aspen [34] and CPAM [33], two state-of-the-art dynamic-graph containers.

Despite these improvements, cache-optimized trees inherently leave performance on the table because they incur random memory accesses rather than reading memory contiguously [12, 63, 83, 85]. Theoretically, cache-friendly trees (e.g., B-trees [10]) are asymptotically optimal in the classical external-memory model [3] for both updates and scans. Empirically, however, tree-based data structures are over 2× slower to scan compared to array-based data

[1]The full version of the paper can be found on arXiv at `https://arxiv.org/abs/2305.05055`.
[2]The artifact and library are available at `https://github.com/wheatman/Packed-Memory-Array`.

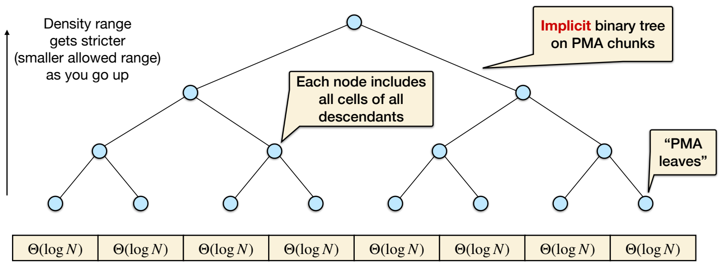

Density range gets stricter (smaller allowed range) as you go up
Implicit binary tree on PMA chunks
Each node includes all cells of all descendants
"PMA leaves"
Θ(log N)

Figure 4: The high-level structure of a PMA and how the single flat array defines an implicit tree.

structures support scans because arrays avoid pointer chasing and therefore take advantage of sequential memory access [63, 79, 85].

Optimizing for sequential access with PMAs The Packed Memory Array (PMA) [47, 13, 14], a dynamic array-based data structure optimized for cache-friendliness (i.e., spatial locality), is a promising candidate for a batch-parallel dynamic-graph container. So far, the PMA has appeared in domains such as graph processing [66, 82, 30, 83, 32, 63, 78], particle simulations [37], and computer graphics [73].

Even though the PMA has appeared in several systems for dynamic graphs, past implementations exhibit low update throughput compared to batch-parallel trees because the PMAs did not include direct algorithmic support for parallel batch updates [83]. Previous work [31] introduced a serial batch-update algorithm for PMAs, but stopped short of parallelization.

Adding batch-parallelism to the Packed Memory Array

PMA structure. The primary feature of a PMA is that it stores its data in sorted order in one contiguous array, which enables fast cache-efficient scans through the elements [47, 13, 14]. To enable efficient updates, the PMA also stores (a constant factor of) empty spaces between its elements to reduce data movement upon updates. Specifically, a PMA with n elements uses $N = \Theta(n)$ cells.

The PMA defines an implicit binary tree which is used during insertions to maintain the proper amount of empty spaces throughout the array. The array is logically (not physically) divided up into leaves of size $\Theta(\log(N))$ cells, so the implicit tree has $\Theta(N/\log(N))$ leaves and height $\Theta(\log(N/\log(N)))$. Every node in the PMA tree corresponds to a ***region*** of cells. Each leaf $i \in \{0,...,N/\log(N)-1\}$ has the region $[i\log(N),(i+1)\log(N))$, and each internal node's region encompasses all of the regions of its descendants. The ***density*** of a region in the PMA is the fraction of occupied cells in that region. Figure 4 illustrates a PMA and its corresponding implicit tree.

Serial inserts in PMAs PMA inserts use the implicit tree to maintain the overall structure. As shown in Figure 5(a), a PMA insert first ***searches*** for the target leaf that the element should go in the sorted order. It then ***places*** the element at the correct location in that leaf. The density bounds guarantee that there is always at least one free cell to place an element in each leaf. Next, it ***counts*** the cells in all necessary nodes in the PMA implicit tree, traversing up until it finds a node that does not violate its density bound. Finally, based on the results from the count, the

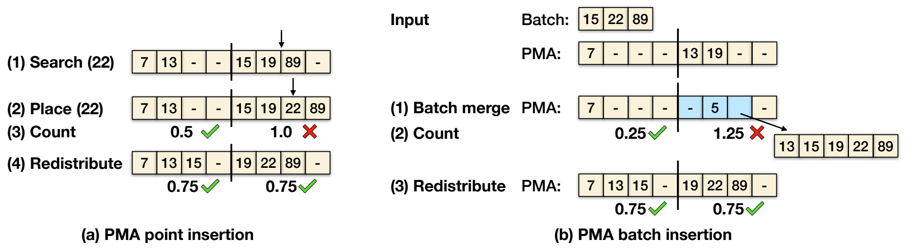


Figure 5: Example of how to perform (a) point inserts and (b) batch inserts in a PMA. The numbers in the count step are the densities of each leaf. In this example, the density bound is 0.9. As shown in step (1), the batch-merge step may overflow a leaf if there are not enough free cells. In that case, elements are stored out of place until the redistribute phase. The full paper contains more details about all steps of the batch-insert algorithm.

PMA ***redistributes*** elements equally among leaves in the node it counted up to, resolving the density bounds in all of its descendants.

Parallel batch-update algorithm for PMAs. The ***batch-insert***[3] ***algorithm*** for PMAs takes as input a PMA with n elements and a batch with k sorted elements to insert. An unsorted batch can be converted into a sorted batch in $O(k\log(k))$ work.

The batch-insert algorithm for PMAs is optimized for the case when the batch is neither too small nor too large. At one extreme, if k is very small (e.g., $k < 100$), the overheads from the batch-insert algorithm outweigh the benefits, so point updates are more efficient than batch updates. At the other extreme, if k is large (e.g., $k \geq n/10$), the optimal algorithm is to rebuild the entire data structure with a linear two-finger merge. The batch-insert algorithm for PMAs performs local merges to address the intermediate case between these two extremes.

At a high level, the batch-insert algorithm for PMAs relies on recursive local merges of the batch elements to the correct PMA leaves in the data structure. In addition to the implicit tree for densities, the PMA also defines another implicit binary search tree on the PMA leaves, where each node is one leaf in the tree (as opposed to the internal nodes encompassing multiple leaves as in the density tree). The recursion begins in the middle leaf of this PMA binary search tree, merges in the appropriate elements from the batch, and in parallel, recurses on the two halves. There are some similarities to batch-insert algorithms for trees, which are implemented with unions/differences [17]. Figure 5(b) provides a worked example of a batch insert in a PMA.

There are two unique challenges to parallel batch updates in PMAs: 1) potentially overflowing leaves with local merges, and 2) efficiently determining which regions to redistribute after the batch merges (via counting). Since multiple leaves may be written to in parallel during the batch merge step, the algorithm cannot overwrite neighboring leaves, even if there are not enough cells to accommodate all elements destined for a given leaf. To address this issue, the batch-insert algorithm for PMAs may store some elements out-of-place temporarily. Furthermore, naively parallelizing the counting step to determine densities (and therefore the regions to redistribute) over the elements in the batch is technically correct because the counting

[3]The batch-parallel PMA supports both inserts and deletes, but we focus on the insert case for ease of presentation. Deletes are implemented symmetrically to inserts, and the full version of the paper contains experiments for both.

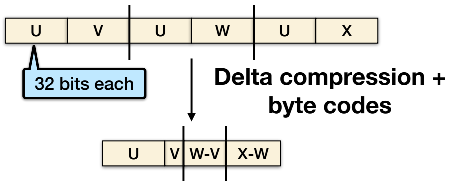


Figure 6: An example of how to store the edges (u,v), (u,w), and (u,x) (assuming $v < w < x$) in F-Graph.

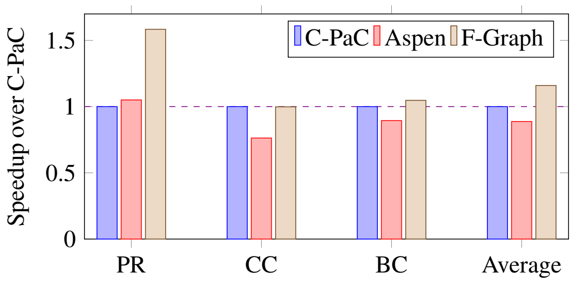


Figure 7: Relative speedup of graph algorithms over C-PaC. The algorithms tested are PageRank (PR), Connected Components (CC), and single-source Betweenness Centrality (BC). The algorithms were tested on a suite of graphs ranging in size from about 86 million edges to about 3.6 billion edges.

is a read-only operation, but may perform a great deal of redundant work. Please see the full paper for details about the batch-insert algorithm, which relies on an efficient parallel counting algorithm to achieve the desired asymptotic bounds.

Using the batch-parallel PMA as a dynamic-graph container

Storing a graph in a compressed flat array F-Graph[4] is built on a single batch-parallel PMA that stores the entire graph as a list of edges in sorted order. It differs from traditional graph representations because it uses only a single array to store both the vertex and edge data. To understand the difference, recall the traditional CSR representation, which stores the graph in two arrays. The offsets array saves space: the edges array then only needs to store destinations and not sources.

As an additional optimization, the PMA underneath F-Graph is compressed with delta compression and byte codes [16, 15, 70]. With delta compression, the Compressed PMA (CPMA) stores differences (deltas) between elements rather than the full elements in all elements except the first in each PMA leaf. Figure 6 illustrates how delta compression saves space by eliding out the source vertex in almost all of the edges (except the start of each leaf and and the first edge of each vertex).

[4]F-Graph uses only one compressed PMA (a flat array) to store the graph. The F in F-Graph comes from the musical key of F, which has one flat.

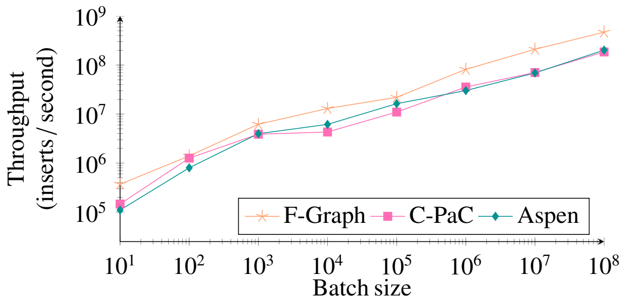


Figure 8: Insert throughput as a function of batch size on the Friendster graph (num. vertices ≈ 125 million, num. edges ≈ 3.6 billion). Edges to add were generated according to an rMAT distribution [20].

Results The empirical evaluation demonstrates that F-Graph is on average 1.2× faster on a suite of graph algorithms, achieves 2× faster throughput for batch updates, and uses marginally less space to store the graphs compared to C-PaC, a state-of-the-art dynamic-graph system based on blocked compressed trees [33]. Furthermore, F-Graph is on average 1.3× faster on graph algorithms, achieves 2× faster throughput for batch updates, and uses 0.6× space to store the graphs compared to Aspen, another high-performance tree-based dynamic-graph system [34].

The full paper includes a much more thorough evaluation, but this article includes a couple of representative plots in Figures 7 and 8. All experiments were run on a 64-core 2-way hyper-threaded Intel machine with 256 GB of memory from AWS [5].

The empirical results demonstrate the potential for dynamic-graph containers to overcome the query-update tradeoff by optimizing for spatial locality. Although cache-optimized trees theoretically dominate[5] PMAs on inserts and match PMAs on scans, in practice, trees are slower to scan because of pointer indirections. The basic theoretical models do not capture the cost of these random accesses, but they have a significant effect on empirical performance. Furthermore, the PMA's cache-friendliness enables F-Graph to supports batch updates much faster than the theoretical bound suggests and even faster than cache-optimized trees because of its locality[6].

Discussion

The empirical advantage of F-Graph, a PMA-based dynamic-graph container, over C-PaC and Aspen, two tree-based dynamic-graph containers, demonstrates the importance of optimizing parallel data structures for the memory subsystem. Specifically, the CPMA's array-based layout enables it to take advantage of the speed of contiguous memory accesses. Despite the theoretical prediction, the batch-parallel CPMA empirically overcomes the query-update tradeoff due to its locality.

[5]Given a cache-line size B and n elements, PMAs support inserts in $\Omega(\log(n)+\log^2(n)/B)$ cache-line transfers, while B-trees support inserts in $O(\log_B(n))$ cache-line transfers in the external-memory model [3].

[6]In microbenchmarks, the PMA was shown to have many fewer L1 and L3 misses when compared to cache-optimized trees.

3 Fair and comprehensive benchmarking of graph containers

The previous sections gloss over an important question in end-to-end system development - *how does the overall dynamic-graph system implement and run the algorithms?* Let us take a step back to examine overall dynamic-graph system performance, since the goal is to efficiently perform both algorithms and updates on the graph. The choice of container is a key decision for holistic system performance, but it is not the only factor.

Structure of dynamic-graph systems. General graph-processing systems have two main components: the ***programming framework*** and ***graph container***. The container stores the graph topology and handles changes to the graph, while the programming framework uses an Application Programming Interface (API), or a specification for how two system components communicate with each other, provided by the container to express and perform analytics.

So far, we have focused on optimizing the graph container in the previous sections, but the programming framework is an equally important factor in graph-algorithm performance. Since both components are necessary for good performance, significant research effort has been devoted to developing and optimizing both sides. On the graph framework side, researchers have developed many high-performance abstractions, such as Ligra [69], the Graph Based Benchmark Suite (GBBS) [35], and the GraphBLAS [28, 27, 51]. Previous work has shown that these abstractions can achieve competitive performance with hand-optimized implementations such as those from the GAP benchmark suite [11].

Issues with current methods for creating dynamic-graph systems

Although there has been great progress on developing and optimizing both the programming framework and graph container, these two directions have mostly been independent lines of work. An ideal dynamic-graph system would combine advancements in both dynamic-graph programming frameworks and containers, as shown in Figure 9. However, integrating different components is challenging because often the framework and container implementations are tightly coupled.

Comprehensiveness of system. The separation between framework and container development results in systems that are limited in terms of performance, capabilities, and generality.

On the frameworks side, for example, the Ligra/GBBS/GraphBLAS abstractions express algorithms in terms of basic data-access primitives to build algorithms that could be implemented by any data structure, but the current codebases use CSR as their representation for simplicity. Although CSR enables good graph-algorithm performance, as we have discussed, it does not support updatability. Furthermore, there are lines of research focused on developing incremental [23, 60, 49, 59, 68, 58] and dynamic [65, 76, 57, 56, 2, 21, 48, 41] algorithm frameworks, but these also implement ad-hoc data structures underneath the framework, leaving performance on the table.

On the container side, systems that include a new dynamic-graph container often run algorithms with either 1) direct implementations of kernels on top of the container [32, 4, 46, 38, 82] or 2) ad-hoc implementations of frameworks [63, 80, 83, 34, 33]. Direct implementations can achieve good performance but limit system generality, since adding new algorithms requires

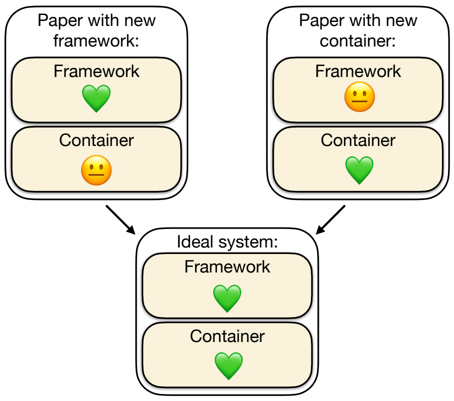


Figure 9: A cartoon to illustrate an ideal-dynamic graph system that combines advances in both dynamic-graph frameworks and containers.

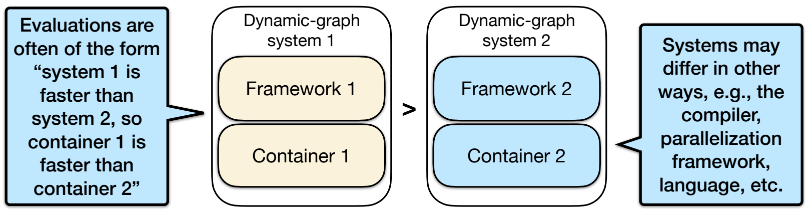

Evaluations are often of the form "system 1 is faster than system 2, so container 1 is faster than container 2"
Dynamic-graph system 1
Framework 1
Container 1
>
Dynamic-graph system 2
Framework 2
Container 2
Systems may differ in other ways, e.g., the compiler, parallelization framework, language, etc.

Figure 10: A cartoon to illustrate the current practice of evaluating end-to-end dynamic-graph systems. Even if a paper claims to introduce a new framework or container, they still must implement the other component, which can affect the overall system performance.

integrating the implementation with each container individually. As a result, systems that use direct implementations usually compare containers on a relatively small set of algorithms. Ad-hoc frameworks enable more general evaluations, but they are not as optimized or expressive compared to implementations that focus on the framework.

Benchmarking graph containers alone. Since the graph-algorithm implementation (either direct or via a framework) and container are usually highly intertwined, most if not all papers introducing new dynamic-graph containers perform *end-to-end* comparisons with existing systems. Although these comparisons aim to compare the containers because the only novelty is in the container, there may be counfounding factors from the rest of the system. As a concrete example, prior evaluations of the dynamic-graph systems SSTGraph [78] and CPAM [33] compare with earlier dynamic-graph systems such as Aspen [34], but change not only the container but also important graph-algorithm details, making the source of any measured improvements unclear. Figure 10 illustrates a high-level view of the difficulty when making container comparisons from end-to-end systems evaluations.

Significant research effort has been devoted to developing and benchmarking graph containers and their corresponding systems. These works have reported significant speedups:

- SSTGraph finds a 1.6× speedup over Aspen [78].

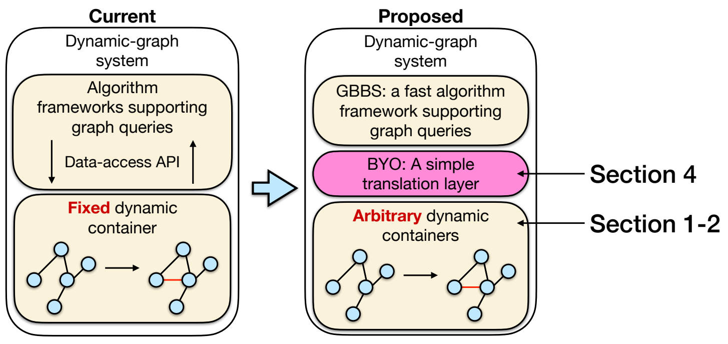

Current
Dynamic-graph system
Algorithm frameworks supporting graph queries
Data-access API
Fixed dynamic container
Proposed
Dynamic-graph system
GBBS: a fast algorithm framework supporting graph queries
BYO: A simple translation layer
Arbitrary dynamic containers
Section 4
Section 1-2

Figure 11: Relationship between graph-algorithm frameworks and dynamic-graph containers in graph-processing systems. BYO provides a simple translation layer between frameworks and containers to enable arbitrary containers to easily plug into the framework.

- Terrace finds a 1.7−2.6× speedup over Aspen [63].
- Aspen finds 1.8−15× speedup over prior dynamic data structures [34].
- VCSR finds as 1.2−2× speedup over PCSR [4].
- PPCSR finds a 1.6× speedup over Aspen [83].
- CompressGraph finds a 2× speedup over Ligra+ [22]
- Teseo finds frequent speedups of at least 1.5× over other graph containers [32]

However, it is likely that much of the improvements seen in these works are from factors other than the graph container itself. A reported performance gain in a proposed dynamic-graph system may be the result of many factors: the container might be better, the system may have better algorithm implementations, or the system may use a better language, compiler, or parallelization library (e.g., pthreads [62], OpenMP [26], Cilk [18], etc.).

As a result, despite the impressive body of existing work on dynamic-graph systems and containers, at present it is essentially impossible to answer the very basic question of *which* container is appropriate for a given graph application.

4 BYO: A unified framework for benchmarking large-scale graph containers

To address these issues, this section describes Bring Your Own (BYO)[7], a unified programming framework for benchmarking and evaluating graph containers [81]. BYO is based on the Graph Based Benchmark Suite (GBBS) [36, 35], a high-performance graph-algorithm framework implemented on top of a CSR container.

Summary

BYO provides a minimal translation layer between GBBS and graph containers (e.g., Aspen, SSTGraph, etc.). In other words, BYO introduces a simple and abstract container API, i.e., the

[7]The full paper is available at `https://arxiv.org/abs/2405.11671`.

API that the containers need to implement, and implements the popular Ligra/GBBS interface[8] using this API. This enables users to easily bring their own graph container and connect it to the programming framework (it suffices if the container implements BYO's container API), as well as to study their own new algorithms (as long as they are expressed in the Ligra/GBBS interface). Figure 11 shows the relationship between BYO, the programming framework, and graph container.

The paper uses BYO to perform a *comprehensive and fair* benchmark of 27 different graph containers, which include both state-of-the-art data structures such as CPAM [33] and SST-Graph [78], as well as off-the-shelf data structure libraries such as those from the `std` standard library and Abseil [1], an open-source standard library from Google. These generic data-structure libraries provide a reference implementation and demonstrate how much performance is left on the table with simple structures and minimal programming effort. The resulting evaluation involves running 10 fundamental graph algorithms on 10 large graph datasets with up to 4.2B edges.

To truly evaluate two graph containers in an apples-to-apples way, BYO ensures that the framework and all other infrastructure (i.e., parallelization library, language, compiler) is consistent across all benchmarks. While this is a seemingly natural requirement, it was not fully met in existing papers evaluating graph systems.

Simplified graph-container evaluation. The graph container API defined by BYO is very simple. Specifically, to implement a wide variety of the primitives in GBBS, *all the graph container developer needs to implement is the map primitive* (excluding basic query functions such as `num_vertices` or `num_edges`). ***Map*** is a functional primitive that applies an arbitrary function `f` over a collection of elements. As we shall see, setting different functions in a map can express other functionality such as reduce and count. A map can easily be implemented with basic iterators such as those in the `C++` standard template library (STL) [61, 50] by applying the function `f` to each element in turn. This feature of BYO greatly simplifies the process of including a new graph container in the benchmark.

For comparison, the graph container API (that the container must support) from GBBS defines 10 primitive neighborhood operations (e.g., map, reduce, scan, etc.). Similarly, the GraphBLAS specification [29] includes 12 operations (e.g., mxm, assign, apply, etc.) for representing graph algorithms.

The main technical challenges in BYO were 1) identifying the correct minimal APIs that can generalize to large classes of graph containers and algorithms, 2) identifying all the code in the original GBBS implementation that makes assumptions about the underlying container and converting them to use modern C++ features that can determine which container functionality to use at compile-time to maximize performance, and 3) simplifying the design to make the translation smooth from the container-developer's point of view.

We built BYO based on GBBS because GBBS has been shown to support a wide variety of theoretically and practically efficient graph algorithms with better performance than alternatives. The full paper verifies these results and shows that BYO achieves $1.06 - 4.44\times$ speedup on average compared to other frameworks (e.g., Ligra [69] and GraphBLAS [51, 19, 27, 28]).

Standardized graph-container evaluation. Furthermore, BYO addresses previous evaluation issues due to different framework implementations by making sensible optimizations for

[8]GBBS is an iteration of Ligra with a richer interface and more algorithm implementations.

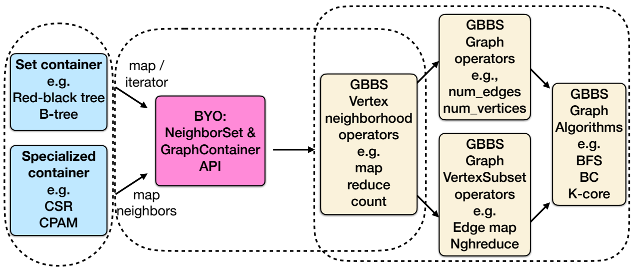


Figure 12: Relationship between BYO framework, graph containers, and graph algorithms (via GBBS).

graph-algorithm performance accessible to all containers that use BYO. For example, the authors of the SSTGraph graph container implemented the Ligra framework on top of SSTGraph [78] to compare with Aspen [34], which also implements Ligra. However, the Ligra implementation in SSTGraph contains additional optimizations for certain algorithms that enable the overall system to achieve better performance on certain workloads. Specifically, SSTGraph found that one of these optimizations helped by 20% on Pagerank and 6% on Connected Components [78]. These optimizations are localized in the programming framework and could theoretically be applied to any dynamic-graph container; by incorporating them, we believe that BYO is the first system that can fairly and reliably isolate performance improvements to the graph container.

BYO API Description

The goal of BYO is to make it as easy as possible for a graph-container developer to use any data structure in a high-performance and general graph programming framework. I will summarize the "set" and "graph container" APIs that BYO exposes to connect with arbitrary graph data structures as well as framework-level optimizations that have appeared in various places throughout the literature that we have collected in BYO. The full paper contains further details about the API components (including discussion of how updates are included in BYO's API) and BYO's implementation.

Figure 12 illustrates the relationship between data structures, BYO, and GBBS components. The GBBS framework uses the VertexSubset abstraction from Ligra [69] for maintaining the active vertex set.

NeighborSet API description. A graph can be represented as a sequence of sets where each set contains the edges incident to a single vertex, that is a neighbor set. Many graph representations, such as the adjacency list in Stinger [38], the tree of trees in Aspen [34] and CPAM [33][9], directly implement this two-level structure.

BYO provides the ***NeighborSet API***, a high-level description of the necessary functionality for *a single* vertex neighbor set. The NeighborSet API enables easy parallelization over the vertex set, since all of the neighbor sets are independent.

[9]The previous section referred to CPAM as C-PaC for clarity compared to CPMA. In this section, we will use CPAM to refer to the library that implements PaC-trees.

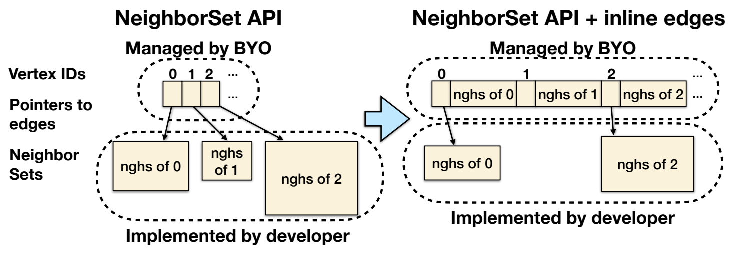


Figure 13: Data structure and inline optimization using the set API.

Figure 13 illustrates the relationship between the vertex level (maintained by BYO) and the set data structures (implemented by the developer). BYO abstracts away the details of choosing a data structure for both the vertex sequence and neighbor sets and enables the user to just implement the neighbor set. Currently, BYO implements the vertex sequence as an `std::vector` for simplicity, but could theoretically use any set data structure.

We find that the minimal API necessary for a neighbor set data structure to implement the GBBS operators which do not change the neighbor sets is to expose a `size` function and an `iterator` which supports sequentially iterating through the elements one by one, i.e., a ***forward*** iterator. These are two basic functionalities are naturally expected from set implementations. For example, the C++ standard template library (STL) [61, 50], a widely-used standard library of basic utilities, includes both of these (among others).

More formally, the required functionality for algorithms for the NeighborSet API is as follows:

- `iterator` or `map(f)`: Apply the function `f` to all elements in the set. An iterator can be used to implement `map` by simply iterating through all elements in the set and applying the function `f`.
- `size()`: Return the number of elements in the set.

Advantages of the NeighborSet API The NeighborSet API is designed to make it as easy as possible for a data-structure developer to integrate their container with BYO, as long as they implement the basic contract specified in the STL container API. Notably, if a developer wants to integrate a set library that implements `size` and `iterator` functionality with BYO, they *do not need to write any additional code*. To improve ease of use, we implemented BYO to automatically translate from the STL container API to the BYO API. That is, integrating a data structure that implements the STL container API just requires importing it at the top of the test driver and specifying its type as the graph container under test.

Furthermore, we incorporate the ***inline optimization*** from Terrace [63] into the vertex set in BYO to benefit arbitrary data structures and enable faster systems overall. This optimization stores a few (about 10) edges inline in the vertex level next to the pointer to the neighbor set for each vertex. The goal is to avoid indirections for low-degree vertices. The idea was originally introduced in Terrace but can be generally applied to any graph container with the sequence of sets structure. Figure 13 illustrates the inline optimization in an arbitrary sequence of sets graph representation. On average, we find that the inline optimization speeds up set containers by 1.06× on average.

GBBS Vertex Operator	*B.Y.O. Lambda*
Map	Pass through provided function
Reduce	auto value = identity map([&](auto ...args) { value.combine(f(args...)) })
Count	int cnt = 0 map([&](auto ...args) { cnt += f(args...) })
Degree	int cnt = 0 map([&](auto ...args) { cnt += 1) })

getNeighbors	Set ngh = { }
	map([&](auto ...args) { ngh.add(args) })

Filter	Set ngh = { }
	map([&](auto ...args) { if (pred(args) ngh.add(args) })

Table 1: GBBS primitives implemented using just the map primitive.

GraphContainer API. Furthermore, BYO provide an alternative ***GraphContainer API*** to connect BYO to graph data structures that do not represent the neighbor sets as separate independent data structures. For example, the classical Compressed Sparse Row (CSR) [72] representation stores all of the neighbor sets contiguously in one array. Furthermore, some optimized dynamic-graph containers like Terrace [63] and SSTGraph [78] collocate some neighbor sets for locality. These graph data structures internally manage both the vertex and neighbor sets.

We find that the minimal API necessary for a data structure to support a diverse set of graph algorithms via BYO is just `map_neighbors` and `num_vertices`:

- `map_neighbors(i, f)`: Apply the function `f` to all neighbors of vertex `i`.
- `num_vertices()`: Return the number of vertices in the graph.

Advantages of the GraphContainer API The GraphContainer API enables cross-set optimizations that cannot be expressed in the NeighborSet API at the cost of programming effort. For example, in the classical CSR, the edges are stored contiguously in one array for locality rather than in separate per-vertex arrays, which is not easily captured by the set of sets abstraction. Another example is the hierarchical structure in Terrace [63], which stores some neighbor sets contiguously in a dynamic array-like data structure. Additionally, SSTGraph [78] shares some metadata between the different neighbor sets for space savings, which cannot be captured with the independent sets abstraction. However, the GraphContainer API cannot access the general inline optimization supported by the NeighborSet API.

Connecting BYO to GBBS BYO simplifies the list of original read-only GBBS neighborhood operators such as map, reduce, count, etc. by implementing several of them with `map`. The original GBBS specification required the data-structure developer to implement several neighborhood operators. In contrast, BYO requires them to implement **only one**. Table 1 demonstrates how to implement the original GBBS neighborhood operators using different `map` lambdas.

In addition to providing the translation layer from the GBBS vertex neighborhood operators, the BYO implementation also modifies the implementation of some operators in GBBS because they assume that the underlying graph is stored in CSR format. This is not inherent in the high-level GBBS specification, but was a prevalent assumption in the codebase. Specifically, several EdgeMap functions assumed that they could directly perform array access into the container to access relevant parts of the graph, which does not hold for arbitrary data structures.

Finally, to further standardize the evaluation between graph containers, BYO includes several framework-level optimizations that can benefit all systems, since they are independent of the container. The paper contains the full details of these optimizations and their performance benefits.

Results

The full paper includes a cross-cutting evaluation of graph containers and frameworks along several distinct axes. I will focus on the graph-container evaluation that BYO enables, but the paper includes benchmarks regarding the BYO framework itself and its comparison to other state-of-the-art frameworks such as Ligra and GraphBLAS. All experimtns were run on an Intel machine with 64 physical cores (128 hyperthreads) and 1024 GiB of main memory.

Using BYO[10], compare over 20 different containers in an apples-to-apples way on graph-algorithm performance without external factors from the algorithm implementation such as the specific algorithm for a problem or systems-level factors like the language and parallelization method. The evaluation includes not only specialized dynamic-graph containers, but also "off-the-shelf" data structures (e.g., those from the standard library) to determine how much performance can be gained just by using simple existing data structure implementations.

The algorithms included in the evaluation cover a wide range of problems, including shortest-path, connectivity, substructure, covering, and eigenvector problems. We refer the interested reader to the GBBS paper for full details on the algorithms and their implementations [35].

The evaluation also measures the performance of dynamic-graph containers when performing batch edge insertions and deletions. BYO also integrates numerous off-the-shelf containers (e.g., Abseil flat hash sets and B-trees), providing a more nuanced picture of dynamic graph containers built using standard data structures that to the best of our knowledge is absent in prior evaluations.

Summary. In terms of graph algorithm performance (Figure 14), our findings show that graph data structures are very similar on average, but that developing specialized graph data structures is worthwhile because additional optimization effort can improve holistic performance on more challenging instances, e.g., high-degree graphs. All of the data structures tested besides the unoptimized `std::set` and `std::unordered_set` incur most about 1.5× slowdown compared to CSR when averaging across all algorithms and graphs. Furthermore, the best specialized container (CPAM with inline) is only about 1.1× faster than the best off-the-shelf data structure (`absl::btree_set` with inline) on average. However, the worst-case slowdown for the `absl::btree_set` is 2.6×, while CPAM achieved a maximum slowdown of 1.9× over CSR. These results suggest that specialized data structures can improve upon off-the-shelf data structures on more difficult problem settings.

BYO cuts through combinatorial explosion in terms of programming effort to enable large-scale comparisons of graph containers on a diverse suite of algorithms to provide a complete view of how fast a graph container can support algorithms in a variety of cases.

In terms of batch inserts (Figure 15), we find that off-the-shelf structures exhibit a folklore query-update tradeoff: the Abseil B-tree, which is best off-the-shelf structure for algorithms,

[10]The code is available at `https://github.com/wheatman/BYO`.

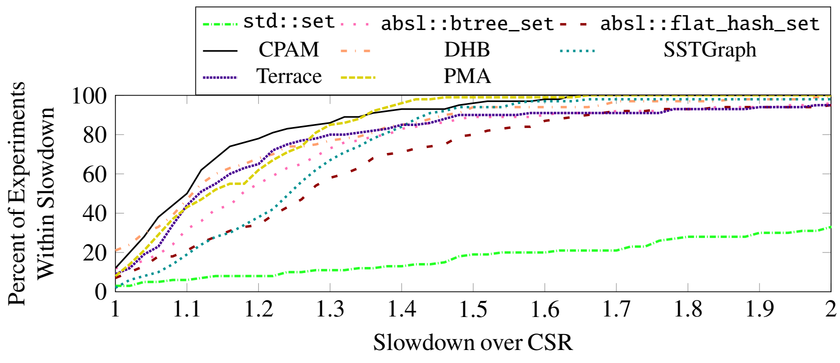


Figure 14: Slowdown of each container compared to CSR. Each point (x,y) for a given graph container means that the container was at most x times slower than CSR on y% of experiments. A line going up faster implies that the container achieves closer performance relative to CSR on more experiments. We find that almost all structures are able to perform the majority of the experiments with at most a 1.4× slowdown over CSR. `std::set` and absl::* are off-the-shelf containers, while the others are specialized for dynamic graphs.

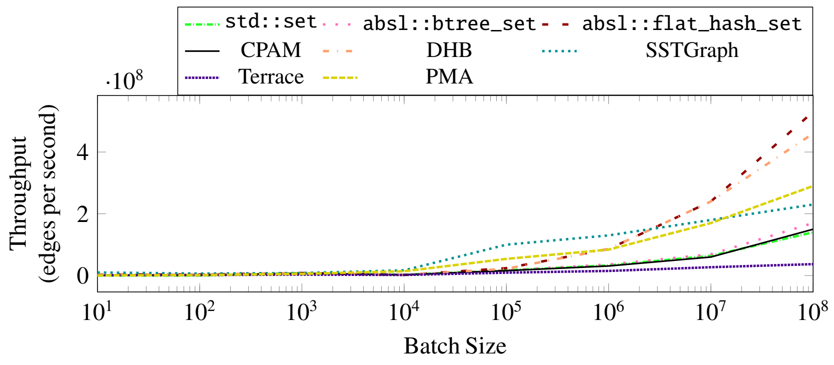


Figure 15: The throughput of inserts for different batch sizes. Of the data structures in this plot, `std::set` and absl::*, are off-the-shelf containers, while the others are specialized for dynamic graphs.

experienced around a 3× slowdown on larger batch inserts compared to the Abseil flat hash set. However, the hash set was worse on algorithms compared to the B-tree. However, specialized containers can overcome the query-update tradeoff on the largest batches: the single PMA is better on algorithms on average compared to the B-tree as well as on the largest batch size.

Comprehensive container evaluation. At a high level, the tested graph data structures are very similar on average, but specialized data structures have an advantage over off-the-shelf structures in terms of worst-case performance across problem instances. Table 2 reports the average, 95th percentile, and maximum slowdown over CSR for each data structure across all 100 problem settings (10 algorithms × 10 graphs).

Container	*Slowdown over CSR*			*Bytes per edge*		
	Average	*95%*	*Max*	*Min*	*Average*	*Max*
NeighborSet API (Vector of...)						
`absl::btree_set`	1.26	1.9	2.3			
`absl::btree_set` (inline)	1.22	2	2.6			
`absl::flat_hash_set`	1.40	2.3	3.4			
`absl::flat_hash_set` (inline)	1.29	2.1	2.6			
`std::set`	2.59	5.0	5.8			
`std::set` (inline)	2.37	4.9	5.6			
`std::unordered_set`	2.01	3.7	6.0			
`std::unordered_set` (inline)	1.90	3.5	5.9			
Aspen	1.22	2	2.5	5.7	12.0	53.4
Aspen (inline)	1.14	1.7	2.0	5.8	7.4	14.9
Compressed Aspen	1.44	2.1	2.6	3.4	5.0	12.1
Compressed Aspen (inline)	1.34	1.9	2.6	3.4	5.5	14.9
CPAM	1.16	1.4	1.5	4.1	4.9	9.0
CPAM (inline)	1.11	1.5	1.6	4.1	6.6	21.6
Compressed CPAM	1.37	1.7	1.9	3.4	4.5	8.9
Compressed CPAM (inline)	1.30	1.8	2.1	3.5	6.2	21.6
PMA	1.25	1.9	3.2	8.1	13.9	46.5
Compressed PMA	1.35	1.9	3.3	4.9	11.2	46.5
Tinyset	1.27	1.9	5.1	5.5	8.6	26.5
Vector	1.07	1.4	1.9	4.1	5.0	10.2
GraphContainer API						
CSR	1.00	1.0	1.0	4.1	5.1	10.6
Compressed CSR	1.23	1.5	1.6	2.3	3.8	10.6
DHB	1.15	1.7	2.4			
PMA	1.15	1.4	1.6	10.0	12.3	24.2
Compressed PMA	1.31	2.0	2.2	3.1	5.6	17.7
SSTGraph	1.25	1.5	2.4	4.0	6.4	19.9
Terrace	1.20	2.0	3.3	9.3	17.7	47.8

Table 2: Data structure algorithm performance and space usage. All data structures are uncompressed unless otherwise specified. Each container's time is normalized to CSR's time averaged over all 100 settings of 10 algorithms × 10 graphs. A number closer to 1 means better performance (higher is worse). The 95% and max columns show the 95th percentile and maximum slowdown over CSR across all algorithms and all graphs. We also show the space usage of the different graph data structures in terms of bytes per edge.

On average, we find that the overall difference between the best off-the-shelf dynamic structure and the best specialized dynamic structure is within about 1.1×. Specifically, the Abseil [1] B-tree incurs 1.22× slowdown compared to CSR. Furthermore, we find that the best specialized graph data structure on average is a vector of uncompressed PaC-trees [33] + inline, which incurred 1.11× slowdown relative to CSR.

The average differences between specialized structures are much smaller than previously reported in other papers because BYO standardizes the evaluation and makes optimizations previously available in one system accessible to all data structures. Specifically, we find that the specialized containers (PaC-trees [33], Terrace [63], DHB [75], CPMA [80], SSTGraph [78] and Aspen [34]) incur between $1.11-1.44\times$ slowdown on average relative to CSR.

These results do not invalidate previous evaluations because BYO enables direct comparisons of *containers* directly rather than overall *systems*. Previously, papers that introduced containers were only able to compare their systems (both the container and framework) because of the lack of a unified easy-to-use framework. Therefore, previously-reported performance differences were the result of variations in the framework as well as the container.

Although the off-the-shelf and specialized data structures achieve similar performance on average, the specialized data structures have better overall performance when looking at the holistic set of experiments. Figure 14 shows for how many experiment settings a given data structure achieved within some slowdown relative to CSR. For example, Abseil's B-tree with the inline optimization achieved within $1.25\times$ of CSR's performance on 63 experiments, while PaC-trees with the inline optimization achieved within $1.25\times$ of CSR's performance on 83 experiments.

We also measure the space usage of all containers which support getting their memory usage in Table 2. We find the bytes per edge varies significantly between graphs even when the container is fixed - by at least $2\times$ and sometimes up to $10\times$. In all cases, the worst-case bytes per edge is on the road graph due to its low degree. Finally, compressed data structures can reduce the space usage by $2\times$ compared their uncompressed counterparts.

Guidance for choosing graph containers. Table 3 shows the fastest container for each combination of graph and algorithm tested. These results provide guidance for choosing among containers for different graph and algorithm types. Please see the paper for details on the algorithms and graphs.

Overall, we find that the optimized tree-based containers (CPAM and Aspen) achieve the best performance most frequently on different problem settings. CPAM performs especially well on the tested Erdos-Renyi (ER) graph [40] graph - it is the fastest on 7/10 algorithms. We conjecture that its performance is due to the uniform degree distribution and relatively high average degree in ER.

Several other containers exhibit strengths in specific algorithm or graph categories:

- The PMA is the fastest container on the *Coloring* algorithm for all graphs. Coloring is a covering-type algorithm that requires iterating over the entire graph in any order, which the PMA is well-suited due to being optimized for contiguous memory access.
- DHB achieves the best performance more often on *large graphs*. We conjecture that DHB is well-suited to large graphs because it uses custom memory allocations that enable it to store more data contiguously.
- Terrace has the best performance on some algorithms (Approximate Densest Subgraph (ADS), Maximal Independent Set (MIS), and PageRank (PR)) when run on the RMAT graph (RM). Terrace is optimized for skewed graphs, and RMAT is a synthetic skewed graph.
- On very sparse graphs, e.g., RD, data structures with fewer pointers and co-located memory such as PMA, CPMA, and SSTGraph are the best choice due to the improved locality of these data structures when the average degree of the graph is extremely low.

To summarize, CPAM and Aspen are solid choices for overall performance, but if a user has a specific algorithm or graph class that they are optimizing for, the trends noted above can help them select a different container.

	RD	*LJ*	*CO*	*RM*	*ER*	*PR*	*TW*	*PA*	*FS*	*KR*
BFS	CPMA	CPAM*	Aspen*	CPAM*	CPAM*	TinySet	CPAM*	Aspen*	CPAM	CPMA
BC	absl::FHS*	CPAM*	Aspen*	CPAM*	CPAM*	DHB	Aspen*	Aspen*	Aspen*	CPAM*
Spanner	PMA	CPAM*	CPAM*	Aspen*	CPAM*	Aspen*	DHB	Aspen*	DHB	DHB
LDD	PMA	CPAM*	CPMA	CPAM*	CPAM*	DHB	CPMA	CPAM*	CPMA	CPMA
CC	absl::FHS*	Aspen*	Aspen	Aspen	Aspen	Aspen	Aspen	DHB	DHB	DHB
ADS	SSTGraph	TinySet	SSTGraph	Terrace	CPAM	absl::btree	PMA	DHB	DHB	DHB
KCore	C-CPAM*	C-CPAM*	CPAM	SSTGraph	SSTGraph	TinySet	CPAM	CPAM*	CPAM*	Aspen*
Coloring	PMA	PMA	PMA	PMA	PMA	PMA	PMA	PMA	PMA	PMA(V)
MIS	PMA	CPAM*	TinySet	Terrace	CPAM	TinySet	Aspen*	CPAM*	Aspen*	Aspen*
PR	DHB	Aspen*	Aspen	Terrace	CPAM*	Aspen	Aspen*	TinySet	PMA (V)	DHB

Table 3: The fastest container for every graph × algorithm combination. The graphs are sorted left to right by size (in number of edges). * next to a container denotes the NeighborSet API version with the inline optimization. CPAM/C-CPAM refers to the uncompressed/compressed version of CPAM, respectively. PMA(V) refers to the vector of PMAs using the NeighborSet API, and PMA refers to the single PMA using the GraphContainer API.

Discussion

This section summarizes BYO, an easy-to-use, high-performance, and expressive graph-algorithm framework. BYO enables apples-to-apples comparisons between dynamic-graph containers by decoupling the graph containers from algorithm implementations. The BYO interface is simple, enabling comprehensive comparisons of new containers on a diverse set of applications with minimal programming effort.

The results demonstrate that the differences between graph containers are smaller than what is commonly reported in papers introducing new graph containers. We attribute this discrepancy to the fact that these papers often perform end-to-end comparisons between graph systems, which vary both the framework and the container. Moreover, the results demonstrate that while on average off-the-shelf data structures achieve highly competitive performance with specialized data structures.

However, the results indicate two promising directions for graph-container developers for optimizing specialized containers. First, the results demonstrate that off-the-shelf structures leave significant performance on the table for certain algorithms/graphs. Designing specialized containers for hard instances can mitigate worst-case performance. Second, specialized graph containers can overcome the folklore query-update tradeoff with efficient parallelization of batch updates.

5 Conclusion

Due to their ubiquity, dynamic graphs have attracted significant research attention towards creating fast algorithms, frameworks, and data structures for processing them. Dynamic-graph algorithms and data structures present an exciting opportunity for practical algorithm engineering that will be required to scale them to large graphs.

This column focuses on developing and benchmarking dynamic-graph containers and their associated systems on modern multicores, which are characterized by their large main memories, many parallel threads, and steep cache hierarchies. Specifically, I discussed two directions:

- Developing dynamic-graph containers that support both fast algorithms and fast updates without giving up performance in either direction. I used F-Graph, a container based on the Packed Memory Array data structure, as a case study for how cache-optimized data structures can overcome traditional performance tradeoffs.

- Comprehensive and fair benchmarking of dynamic-graph containers without confounding factors from the overall dynamic-graph system, which includes both algorithm framework and the container. I show how BYO, a high-performance graph-algorithm framework designed for simplicity and ease of use, can standardize evaluations of graph containers and enable developers to easily build expressive and fast dynamic-graph systems.

Remarks and future directions

I believe that the two concrete results I mentioned in the column are steps in the right direction, but there is potential for the high-level ideas to play an even wider role in future development and benchmarking of dynamic-graph systems.

The results in this column demonstrate the Packed Memory Array's potential as a replacement for dynamic trees, but the PMA is not yet comparable to trees in terms of functionality and generality. For example, tree implementations often target more complex use cases such as functional updates [34] and transactional updates [25]. Furthermore, trees are common in other settings such as out-of-core and disk-based storage systems, but to my knowledge, there is not yet a high-performance disk-based PMA implementation.

Furthermore, BYO takes the first step towards apples-to-apples comparisons of graph containers, but is built on GBBS, which currently runs static graph algorithms, or algorithms that must recompute the entire answer from scratch in the presence of changes to the graph. There has been a great deal of work from the theory community on developing dynamic-graph algorithms, or algorithms that maintain some intermediate state to take updates into account and avoid full recomputation. I refer the interested reader to an excellent survey on the topic [44]. However, it is non-trivial to implement dynamic-graph algorithms that are actually faster than static algorithms in practice due to parallelism and locality issues. Independently of the theory community, the systems community has proposed many programming frameworks for dynamic algorithms on graphs [23, 60, 49, 59, 68, 58, 65, 76, 57, 56, 2, 21, 48, 41]. However, these framework implementations are also often tightly coupled with their underlying graph container, so they suffer from the same issues of generality.

My main motivation for writing this column is to share my excitement for developing dynamic-graph data structures and their associated systems. Although there have already been many papers on this topic, I believe that it will continue to be an area for future development as dynamic graphs continue to grow. I look forward to seeing progress in these exciting directions.

Acknowledgements

Special thanks to all my collaborators who made this work possible - Brian Wheatman, Prashant Pandey, Aydın Buluç, Randal Burns, Xiaojun Dong, Zheqi Shen, Laxman Dhulipala, and Jakub Łącki.

References

[1] Abseil. `https://abseil.io/`. Accessed: 2023-09-23.

[2] Mahbod Afarin, Chao Gao, Shafiur Rahman, Nael Abu-Ghazaleh, and Rajiv Gupta. CommonGraph: Graph analytics on evolving data. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS 2023, page 133–145, New York, NY, USA, 2023. Association for Computing Machinery.

[3] Alok Aggarwal and Jeffrey S. Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1127, September 1988.

[4] Abdullah Al Raqibul Islam, Dong Dai, and Dazhao Cheng. VCSR: Mutable CSR graph format using vertex-centric Packed Memory Array. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 71–80, 2022.

[5] Amazon. Amazon web services. `https://aws.amazon.com/`, 2022.

[6] Ariful Azad, Georgios A Pavlopoulos, Christos A Ouzounis, Nikos C Kyrpides, and Aydin Buluç. HipMCL: a high-performance parallel implementation of the markov clustering algorithm for large-scale networks. *Nucleic acids research*, 46(6):e33–e33, 2018.

[7] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54, 2006.

[8] David A. Bader and K. Madduri. High-performance combinatorial techniques for analyzing massive dynamic interaction networks. In *DIMACS Workshop on Computational Methods for Dynamic Interaction Networks, DIMACS Center, Rutgers University, Piscataway, NJ, September 24-25, 2007.*, 2007.

[9] Antonio Barbuzzi, Pietro Michiardi, Ernst Biersack, and Gennaro Boggia. Parallel bulk insertion for large-scale analytics applications. In *Proceedings of the 4th International Workshop on Large Scale Distributed Systems and Middleware*, pages 27–31, 2010.

[10] Rudolf Bayer and Edward M. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, 1(3):173–189, 1972.

[11] Scott Beamer, Krste Asanović, and David Patterson. The GAP benchmark suite. *arXiv preprint arXiv:1508.03619*, 2015.

[12] Michael A Bender, Alex Conway, Martín Farach-Colton, William Jannen, Yizheng Jiao, Rob Johnson, Eric Knorr, Sara McAllister, Nirjhar Mukherjee, Prashant Pandey, et al. Small refinements to the DAM can have big consequences for data-structure design. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, pages 265–274, 2019.

[13] Michael A Bender, Erik D Demaine, and Martin Farach-Colton. Cache-oblivious B-trees. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 399–409. IEEE, 2000.

[14] Michael A Bender, Erik D Demaine, and Martin Farach-Colton. Cache-oblivious B-trees. *SIAM Journal on Computing*, 35(2):341–358, 2005.

[15] D. K. Blandford, G. E. Blelloch, and I. A. Kash. Compact representations of separable graphs. In *SODA*, 2003.

[16] Daniel K. Blandford, Guy E. Blelloch, and Ian A. Kash. An experimental analysis of a compact graph representation. In *ALENEX*, 2004.

[17] Guy E. Blelloch, Daniel Ferizovic, and Yihan Sun. Just join for parallel ordered sets. SPAA '16, pages 253–264, New York, NY, USA, 2016. Association for Computing Machinery.

[18] Robert D Blumofe, Christopher F Joerg, Bradley C Kuszmaul, Charles E Leiserson, Keith H Randall, and Yuli Zhou. Cilk: An efficient multithreaded runtime system. *ACM SigPlan Notices*, 30(8):207–216, 1995.

[19] Aydin Buluç, Tim Mattson, Scott McMillan, José Moreira, and Carl Yang. Design of the GraphBLAS API for C. In *2017 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*, pages 643–652. IEEE, 2017.

[20] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-MAT: A recursive model for graph mining. In *SDM*, pages 442–446, 2004.

[21] Dan Chen, Chuangyi Gui, Yi Zhang, Hai Jin, Long Zheng, Yu Huang, and Xiaofei Liao. Graphfly: efficient asynchronous streaming graphs processing via dependency-flow. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14. IEEE, 2022.

[22] Zheng Chen, Feng Zhang, JiaWei Guan, Jidong Zhai, Xipeng Shen, Huanchen Zhang, Wentong Shu, and Xiaoyong Du. CompressGraph: Efficient parallel graph analytics with rule-based compression. *Proc. ACM Manag. Data*, 1(1), may 2023.

[23] Raymond Cheng, Ji Hong, Aapo Kyrola, Youshan Miao, Xuetian Weng, Ming Wu, Fan Yang, Lidong Zhou, Feng Zhao, and Enhong Chen. Kineograph: taking the pulse of a fast-changing and connected world. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 85–98, 2012.

[24] Yongli Cheng, Yan Ma, Hong Jiang, Lingfang Zeng, Fang Wang, Xianghao Xu, and Yuhang Wu. TgStore: An efficient storage system for large time-evolving graphs. *IEEE Transactions on Big Data*, (01):1–16, 2024.

[25] Douglas Comer. Ubiquitous B-tree. *ACM Computing Surveys (CSUR)*, 11(2):121–137, 1979.

[26] Leonardo Dagum and Ramesh Menon. OpenMP: an industry standard API for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998.

[27] Timothy A. Davis. Algorithm 1000: SuiteSparse:GraphBLAS: Graph algorithms in the language of sparse linear algebra. *ACM Trans. Math. Softw.*, 45(4), dec 2019.

[28] Timothy A. Davis. Algorithm 10xx: SuiteSparse:GraphBLAS: Parallel graph algorithms in the language of sparse linear algebra. *ACM Trans. Math. Softw.*, jan 2023. Just Accepted.

[29] Timothy A. Davis. User Guide for SuiteSparse:GraphBLAS. `https://github.com/DrTimothyAldenDavis/GraphBLAS/blob/stable/Doc/GraphBLAS_UserGuide.pdf`, September 2023.

[30] Dean De Leo and Peter Boncz. Fast concurrent reads and updates with PMAs. GRADES-NDA'19, pages 8:1–8:8, New York, NY, USA, 2019. ACM.

[31] Dean De Leo and Peter Boncz. Packed memory arrays-rewired. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 830–841. IEEE, 2019.

[32] Dean De Leo and Peter Boncz. Teseo and the analysis of structural dynamic graphs. *Proceedings of the VLDB Endowment*, 14(6):1053–1066, 2021.

[33] Laxman Dhulipala, Guy E. Blelloch, Yan Gu, and Yihan Sun. PaC-Trees: Supporting parallel and compressed purely-functional collections. In *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2022, page 108–121, New York, NY, USA, 2022. Association for Computing Machinery.

[34] Laxman Dhulipala, Guy E. Blelloch, and Julian Shun. Low-latency graph streaming using compressed purely-functional trees. In *PLDI*, pages 918–934, 2019.

[35] Laxman Dhulipala, Guy E. Blelloch, and Julian Shun. Theoretically efficient parallel graph algorithms can be fast and scalable. *ACM Trans. Parallel Comput.*, 8(1), apr 2021.

[36] Laxman Dhulipala, Jessica Shi, Tom Tseng, Guy E. Blelloch, and Julian Shun. The Graph Based Benchmark Suite (GBBS). In *Proceedings of the 3rd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, GRADES-NDA'20, New York, NY, USA, 2020. Association for Computing Machinery.

[37] Marie Durand, Bruno Raffin, and François Faure. A packed memory array to keep moving particles sorted. In *9th Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)*, pages 69–77. The Eurographics Association, 2012.

[38] David Ediger, Robert McColl, Jason Riedy, and David A. Bader. Stinger: High performance data structure for streaming graphs. In *High Performance Extreme Computing (HPEC), 2012 IEEE Conference on*, pages 1–5. IEEE, 2012.

[39] Stephan Erb, Moritz Kobitzsch, and Peter Sanders. Parallel bi-objective shortest paths using weight-balanced b-trees with bulk updates. In *International Symposium on Experimental Algorithms*, pages 111–122. Springer, 2014.

[40] Paul Erdös and Alfréd Rényi. On random graphs I. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[41] Guanyu Feng, Zixuan Ma, Daixuan Li, Shengqi Chen, Xiaowei Zhu, Wentao Han, and Wenguang Chen. Risgraph: A real-time streaming system for evolving graphs to support sub-millisecond per-update analysis at millions ops/s. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, page 513–527, New York, NY, USA, 2021. Association for Computing Machinery.

[42] Leonor Frias and Johannes Singler. Parallelization of bulk operations for stl dictionaries. In *European Conference on Parallel Processing*, pages 49–58. Springer, 2007.

[43] Per Fuchs, Domagoj Margan, and Jana Giceva. Sortledton: a universal, transactional graph data structure. *Proceedings of the VLDB Endowment*, 15(6):1173–1186, 2022.

[44] Kathrin Hanauer, Monika Henzinger, and Christian Schulz. Recent advances in fully dynamic graph algorithms–a quick reference guide. *ACM Journal of Experimental Algorithmics*, 27:1–45, 2022.

[45] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[46] Abdullah Al Raqibul Islam and Dong Dai. DGAP: Efficient dynamic graph analysis on persistent memory. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '23, New York, NY, USA, 2023. Association for Computing Machinery.

[47] Alon Itai, Alan G. Konheim, and Michael Rodeh. A sparse table implementation of priority queues. In *ICALP*, pages 417–431, 1981.

[48] Anand Padmanabha Iyer, Qifan Pu, Kishan Patel, Joseph E Gonzalez, and Ion Stoica. TEGRA: Efficient Ad-Hoc analytics on evolving graphs. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 337–355, 2021.

[49] Xiaolin Jiang, Chengshuo Xu, Xizhe Yin, Zhijia Zhao, and Rajiv Gupta. Tripoline: generalized incremental graph processing via graph triangle inequality. In *Proceedings of the Sixteenth European Conference on Computer Systems*, pages 17–32, 2021.

[50] Nicolai M Josuttis. The C++ standard library: a tutorial and reference. 2012.

[51] Jeremy Kepner, Peter Aaltonen, David Bader, Aydın Buluç, Franz Franchetti, John Gilbert, Dylan Hutchison, Manoj Kumar, Andrew Lumsdaine, Henning Meyerhenke, et al. Mathematical foundations of the GraphBLAS. In *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9. IEEE, 2016.

[52] Pradeep Kumar and H Howie Huang. GraphOne: A data store for real-time analytics on evolving graphs. *ACM Transactions on Storage (TOS)*, 15(4):1–40, 2020.

[53] Aapo Kyrola, Guy E. Blelloch, and Carlos Guestrin. Graphchi: Large-scale graph computation on just a pc. USENIX, 2012.

[54] Peter Macko, Virendra J Marathe, Daniel W Margo, and Margo I Seltzer. LLAMA: Efficient graph analytics using large multiversioned arrays. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 363–374. IEEE, 2015.

[55] Kamesh Madduri and David A. Bader. Compact graph representations and parallel connectivity algorithms for massive dynamic network analysis. In *2009 IEEE International Symposium on Parallel & Distributed Processing*, pages 1–11. IEEE, 2009.

[56] Mugilan Mariappan, Joanna Che, and Keval Vora. DZiG: Sparsity-aware incremental processing of streaming graphs. In *Proceedings of the Sixteenth European Conference on Computer Systems*, pages 83–98, 2021.

[57] Mugilan Mariappan and Keval Vora. Graphbolt: Dependency-driven synchronous processing of streaming graphs. In *Proceedings of the Fourteenth EuroSys Conference 2019*, pages 1–16, 2019.

[58] Frank McSherry, Andrea Lattuada, Malte Schwarzkopf, and Timothy Roscoe. Shared arrangements: Practical inter-query sharing for streaming dataflows. *Proc. VLDB Endow.*, 13(10):1793–1806, jun 2020.

[59] Frank McSherry, Derek Gordon Murray, Rebecca Isaacs, and Michael Isard. Differential dataflow. In *CIDR*, 2013.

[60] Derek G Murray, Frank McSherry, Rebecca Isaacs, Michael Isard, Paul Barham, and Martín Abadi. Naiad: a timely dataflow system. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 439–455, 2013.

[61] David R Musser, Gilmer J Derge, and Atul Saini. *STL tutorial and reference guide: C++ programming with the standard template library*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[62] Bradford Nichols, Dick Buttlar, and Jacqueline Farrell. *Pthreads programming: A POSIX standard for better multiprocessing*. " O'Reilly Media, Inc.", 1996.

[63] Prashant Pandey, Brian Wheatman, Helen Xu, and Aydın Buluç. Terrace: A hierarchical graph container for skewed dynamic graphs. In *SIGMOD*, page 1372–1385, 2021.

[64] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[65] Dipanjan Sengupta, Narayanan Sundaram, Xia Zhu, Theodore L Willke, Jeffrey Young, Matthew Wolf, and Karsten Schwan. Graphin: An online high performance incremental graph processing framework. In *Euro-Par 2016: Parallel Processing: 22nd International Conference on Parallel and Distributed Computing, Grenoble, France, August 24-26, 2016, Proceedings 22*, pages 319–333. Springer, 2016.

[66] Mo Sha, Yuchen Li, Bingsheng He, and Kian-Lee Tan. Accelerating dynamic graph analytics on GPUs. *Proc. VLDB Endow.*, 11(1):107–120, September 2017.

[67] Jifan Shi, Biao Wang, and Yun Xu. Spruce: a fast yet space-saving structure for dynamic graph storage. *Proceedings of the ACM on Management of Data*, 2(1):1–26, 2024.

[68] Xiaogang Shi, Bin Cui, Yingxia Shao, and Yunhai Tong. Tornado: A system for real-time iterative analysis over evolving data. In *Proceedings of the 2016 International Conference on Management of Data*, pages 417–430, 2016.

[69] Julian Shun and Guy E. Blelloch. Ligra: A lightweight graph processing framework for shared memory. In *PPoPP*, pages 135–146, 2013.

[70] Julian Shun, Laxman Dhulipala, and Guy E. Blelloch. Smaller and faster: Parallel processing of compressed graphs with Ligra+. In *2015 Data Compression Conference*, pages 403–412. IEEE, 2015.

[71] Yihan Sun, Daniel Ferizovic, and Guy E. Belloch. PAM: parallel augmented maps. In *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 290–304, 2018.

[72] William F. Tinney and John W. Walker. Direct solutions of sparse network equations by optimally ordered triangular factorization. *Proceedings of the IEEE*, 55(11):1801–1809, 1967.

[73] Julio Toss, Cicero AL Pahins, Bruno Raffin, and João LD Comba. Packed-memory quadtree: A cache-oblivious data structure for visual exploration of streaming spatiotemporal big data. *Computers & Graphics*, 76:117–128, 2018.

[74] Thomas Tseng, Laxman Dhulipala, and Guy E. Blelloch. Batch-parallel euler tour trees. In *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 92–106. SIAM, 2019.

[75] Alexander van der Grinten, Maria Predari, and Florian Willich. A fast data structure for dynamic graphs based on hash-indexed adjacency blocks. In *20th International Symposium on Experimental Algorithms (SEA 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[76] Keval Vora, Rajiv Gupta, and Guoqing Xu. Kickstarter: Fast and accurate computations on streaming graphs via trimmed approximations. In *Proceedings of the twenty-second international conference on architectural support for programming languages and operating systems*, pages 237–251, 2017.

[77] Rui Wang, Shuibing He, Weixu Zong, Yongkun Li, and Yinlong Xu. XPGraph: XPline-friendly persistent memory graph stores for large-scale evolving graphs. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1308–1325. IEEE, 2022.

[78] Brian Wheatman and Randal Burns. Streaming sparse graphs using efficient dynamic sets. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 284–294. IEEE, 2021.

[79] Brian Wheatman, Randal Burns, Aydın Buluç, and Helen Xu. Optimizing search layouts in Packed Memory Arrays. In *ALENEX*, 2023.

[80] Brian Wheatman, Randal Burns, Aydın Buluç, and Helen Xu. CPMA: An efficient batch-parallel compressed set without pointers. In *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pages 348–363, 2024.

[81] Brian Wheatman, Xiaojun Dong, Zheqi Shen, Laxman Dhulipala, Jakub Łącki, Prashant Pandey, and Helen Xu. BYO: A unified framework for benchmarking large-scale graph containers. *Proceedings of the VLDB Endowment (to appear)*, 2024.

[82] Brian Wheatman and Helen Xu. Packed compressed sparse row: A dynamic graph representation. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*. IEEE, 2018.

[83] Brian Wheatman and Helen Xu. A parallel Packed Memory Array to store dynamic graphs. In *ALENEX*, pages 31–45, 2021.

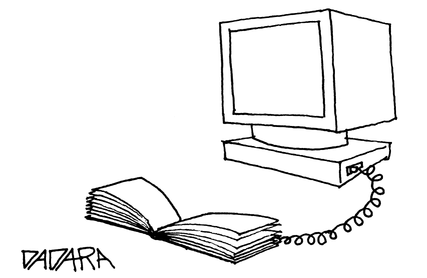
[84] Martin Winter, Daniel Mlakar, Rhaleb Zayer, Hans-Peter Seidel, and Markus Steinberger. faimGraph: High performance management of fully-dynamic graphs under tight memory constraints on the GPU. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 754–766. IEEE, 2018.

[85] Helen Xu, Amanda Li, Brian Wheatman, Manoj Marneni, and Prashant Pandey. BP-Tree: Overcoming the point-range operation tradeoff for in-memory B-trees. *Proc. VLDB Endow.*, 16(11):2976–2989, jul 2023.

[86] Helen Jiang Xu. *The Locality-First Strategy for Developing Efficient Multicore Algorithm*. PhD thesis, Massachusetts Institute of Technology, 2022.

[87] Lei Zou, Fan Zhang, Yinnian Lin, and Yanpeng Yu. An efficient data structure for dynamic graph on GPUs. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

News and Conference Reports



Report from EATCS Japan Chapter

Yuto Nakashima (Kyushu University)

EATCS-JP/LA Workshop on TCS and Presentation Awards

The 22nd *EATCS-JP/LA Workshop on Theoretical Computer Science* was held at Maskawa Hall, Kyoto University, February 19th to 21st, 2024. (The details can also be found, although this website is written in Japanese, at `https://la-symposium.github.io/2023/winter_program.html`.)

Every year, we choose the best presenter and the best student presenter. This year, we celebrated the following presentation as the 22nd LA/EATCS-Japan Presentation Award:

> "*Analysis of Voting Process from Martingale Concentration*", **Nobutaka Shimizu** (Tokyo Institute of Technology) and Takeharu Shiraga (Chuo University)

We celebrated the following presentation as the 13th LA/EATCS-Japan Student Presentation Award:

> "*Counting the Number of Non-overlapping Edge Unfoldings in Convex Regular Faced Polyhedra*", **Takumi Shiota** (Kyushu Institute of Technology), Yudai Enomoto, Takashi Horiyama (Hokkaido University), and Toshiki Saitoh (Kyushu Institute of Technology)

The awards were recognized publicly on the last day, February 21st, 2024.

Congratulations!

This workshop is jointly organized by *LA symposium*, Japanese association of theoretical computer scientists. Its purpose is to give a place to discuss topics on all aspects of theoretical computer science. This workshop is an unrefereed meeting. All submissions are accepted for the presentation. There should be no problem of presenting these papers at refereed conferences and/or journals. This meeting is familiar and widely open for everyone who is interested in theoretical computer science. It is held twice a year (January/February and July/August). If you have a chance, I recommend that you attend it. Check the website `http://www.dais.is.tohoku.ac.jp/eatcs_japan/` for further details. The list of the presentations is as below; you can see the activity of the Japanese society of theoretical computer science.

Program of the 22nd EATCS-JP/LA workshop on TCS

In the following program, "*" indicates speakers. The number [*xx*S] means it is given by student speakers.

[1] A Data Structure for the Maximum-Sum Segment Problem with Offsets
**Yoshifumi Sakai (Tohoku University)*

[2] A Matching Algorithm for Term Tree Patterns with Contractible Variables
**Yusuke Suzuki, Tomoyuki Uchida (Hiroshima City University), Takayoshi Shoudai (Fukuoka Institute of Technology), Satoshi Matsumoto (Tokai University), Tetsuhiro Miyahara (Hiroshima City University)*

[3] Compactness for Finite Unions of Non-adjacent Regular Pattern Languages
*Naoto Taketa, *Tomoyuki Uchida (Hiroshima City University), Takayoshi Shoudai (Fukuoka Institute of Technology), Satoshi Matsumoto (Tokai University), Yusuke Suzuki, Tetsuhiro Miyahara (Hiroshima City University)*

[4S] Optimal Strategy for YOMEN
**Hirano Kouki (Nagoya University), Kiya Hironori (Osaka Metropolitan University), Hanaka Tesshu (Kyushu University), Ono Hirotaka (Nagoya University)*

[5S] Balanced Generalized Janken and its Isomorphism
**Akari Toyonaga, Atsuki Nagao (Ochanomizu University)*

[6S] Analysis of Hex and Doubutsu Shogi Powered by Graphillion
**Kazutaka Aoki, Hiroshi Fujiwara (Shinshu University)*

[7S] Reconfiguration Graphs of Independent Sets under Token Jumping
**Masaya Sano, Hiroshi Fujiwara (Shinshu University)*

[8S] Enumeration Algorithms for Maximum Matchings in Chain Graphs
**Yosei Kamada, Hiroshi Fujiwara (Shinshu University)*

[9S] The Least Core of Routing Games
**Tomohiro Kobayashi, Tomomi Matsui (Tokyo Institute of Technology)*

[10S] Polyhedral Characterization of Pareto Optimal Matchings – Approaches to Unpopularity-Matching Problem –
**Iori Moriyama, Tomomi Matsui (Tokyo Institute of Technology)*

[11S] FPT Approximation Schemes for Connected Graph Partitioning Problems with Cardinality Constraints
**Suguru Yamada, Tesshu Hanaka (Kyushu University)*

[12S] NP-Completeness of Power Domination Set Problem for Planer Graphs
**Daichi Sawayama (Saitama university), Toshinori Yamada (Saitama university)*

[13S] On the Stackelberg Minimum Cost Flow Problem
**Hideki Kitada, Toshinori Yamada (Saitama University)*

[14S] Solving the Bike Sharing Problem for Arbitrary Positioning and Constrained Movement
**Valeri Haralanov, Toshinori Yamada (Saitama University)*

[15S] Recurrence and Transince of Random Walks on Growing Dimensional Boxes
**Shuma Kumamoto (Kyushu University), Shuji Kijima (Shiga University), Tomoyuki Shirai (Kyushu University)*

[16] Reversibility of Finite Cellular Automata on Monoids
**Shuichi Inokuchi, Yuta Arima, Chihiro Iwanaga (Fukuoka Institute of Technology)*

[17S] An Edit Model and Algorithms for Achieving Properties on Intersection Graphs

**Nicolas Honorato Droguet, Kazuhiro Kurita (Nagoya University), Tesshu Hanaka (Kyushu University), Hirotaka Ono (Nagoya University)*

[18S] An Improvement of a Spectral Lower Bound for Treewidth
*Tatsuya Gima (Nagoya University), Tesshu Hanaka (Kyushu University), *Kohei Noro, Hirotaka Ono, Yota Otachi (Nagoya University)*

[19S] Improved Upper Bounds for Treewidth of Outer k-Planar Graphs
*Oksana Firman (Universität Würzburg), Myroslav Kryven (University of Manitoba), *Yuto Okada (Nagoya University), Alexander Wolff (Universität Würzburg)*

[20S] An Efficient Beer Path Query System Based on Graph Decomposition
**Kosuke Sugiyama (Nagoya University), Tesshu Hanaka (Kyushu University), Hirotaka Ono (Nagoya University), Kunihiko Sadakane (The University of Tokyo)*

[21S] Optimal Parameterized Quantum Query Complexity of Vertex Cover and Matching for Small k
**Tatsuya Terao (Kyoto University), Ryuhei Mori (Nagoya University)*

[22S] Random Generation of Tent Codes
**Naoaki Okada (Kyushu University), Shuji Kijima (Shiga University)*

[23] On the Communication Complexity of Non-Interactive Secure Multiparty Computation
**Maki Yoshida (NICT)*

[24] Analysis of Voting Process from Martingale Concentration
**Nobutaka Shimizu (Tokyo Institute of Technology), Takeharu Shiraga (Chuo University)*

[25S] LZ Parsing with Height Bounds
*Hideo Bannai (Tokyo Medical and Dental University), Mitsuru Funakoshi (NTT Communication Science Laboratories), Diptarama Hendrian, *Myuji Matsuda (Tokyo Medical and Dental University), Simon J. Puglisi (University of Helsinki)*

[26S] Computing Longest Border and Shortest Cover After Block Edit
**Kazuki Mitani (Hokkaido University), Takuya Mieno (University of Electro-Communications), Kazuhisa Seto, Takashi Horiyama (Hokkaido University)*

[27S] An Upper Bound on the Number of Maximal α-gapped Repeats in the Fibonacci Words
**Kazuma Yamane (Hokkaido University), Yuto Nakashima (Kyushu University), Kazuhisa Seto, Takashi Horiyama (Hokkaido University)*

[28] Computing Repetitiveness Measures with Answer Set Programming
**Dominik Köppl (University of Yamanashi), Mutsunori Banbara (Nagoya University)*

[29S] NP-Hardness of Minimum-Time Packages Delivery in the Line by Robots with Different Speeds
**Toshiya Hiratsuka, Yamada Toshinori (Saitama University)*

[30] Near-linear Time Dispersion of Mobile Agents
**Yuichi Sudo (Hosei University), Masahiro Shibata (Kyushu Institute of Technology), Junya Nakamura (Toyohashi University of Technology), Yonghwan Kim (Nagoya Institute of Technology), Toshimitsu Masuzawa (Osaka University)*

[31] PLS Is Contained in PLC
**Takashi Ishizuka (Fujitsu Limited)*

[32S] Clinching Auctions with Additional Buyers

**Ryosuke Sato (University of Tokyo)*

[33S] Counting the Number of Non-overlapping Edge Unfoldings in Convex Regular Faced Polyhedra

**Takumi Shiota (Kyushu Institute of Technology), Yudai Enomoto, Takashi Horiyama (Hokkaido University), Toshiki Saitoh (Kyushu Institute of Technology)*

[34S] Computing Diverse Pair of Solutions for SAT

*Tatsuya Gima (Nagoya University), Yuni Iwamasa (Kyoto University), Yasuaki Kobayashi (Hokkaido University), Kazuhiro Kurita, Yota Otachi (Nagoya University), *Rin Saito (Tohoku University)*

[35] On the Hardness of Minimal Steiner Node Multicut Enumeration

*Yasuaki Kobayashi (Hokkaido University), *Kazuhiro Kurita (Nagoya University)*

[36S] Lipschitz Continuous Algorithms for Covering Problems

**Soh Kumabe (The University of Tokyo), Yuichi Yoshida (National Institute of Informatics)*

Past/Forthcoming Events

WALCOM 2024 & 2025

International Conference and Workshops on Algorithms and Computation (WALCOM) conference has been established to encourage the researchers of theoretical computer science in Asia, especially, India and Bangladesh. Nowadays, there are many participants from a wide range of Asia, not so many from Europe so far. The organizers give a big welcome to many attendees from Europe. The 18th WALCOM (WALCOM 2024) was held in Kanazawa, Japan, from March 18th to 20th, 2024. See `https://www.kono.cis.iwate-u.ac.jp/~yamanaka/walcom2024/` for more information on WALCOM 2024. The next WALCOM will be held in Chengdu, China, from February 28th to March 2nd, 2025. See `https://tcsuestc.com/walcom2025/index.html` for more information on WALCOM 2025. The important dates are as follows:

Submission Deadline: September 22, 2024 (Anywhere on Earth)

Notification of Acceptance: November 8, 2024

AAAC 2024

Annual Meeting of the Asian Association for Algorithms and Computation (AAAC) aims at promoting collaborations in theoretical computer science in Asia (but not restricted in the region). The 15th AAAC was held in Osaka, Japan, from May 31st to June 1st, 2024. See `https://cs.kwansei.ac.jp/~tokuyama/AAAC2024.html` for more information on AAAC 2024.

CPM 2024

The 35th Annual Symposium on Combinatorial Pattern Matching (CPM 2024) will be held in Fukuoka, Japan, from June 25th to June 27th, 2024 with summer school on June 20th and 21st in Tokyo and StringMasters workshop on June 25th and 28th in Fukuoka. See `https://cpm2024.github.io/index.html` for more information of these events.

WAAC 2024
The 24th Korea–Japan Joint Workshop on Algorithms and Computation (WAAC 2024) will be held in Seoul, Korea, on August 2nd and 3rd, 2024. The aim of this workshop is to provide a forum for researchers working on algorithms and the theory of computation, to promote the exchange of recent results, to foster new collaborations among researchers. Historically, the workshop has established for the purpose of collaboration of researchers of Korea and Japan; however, participation from any country is welcome. See `https://algo.postech.ac.kr/workshops/waac24/` for more information of WAAC 2024.

CIAA 2024
International Conference on Implementation and Application of Automata (CIAA) is an annual conference that concerns research on all aspects of implementation and application of automata and related structures, including theoretical aspects. The 28th CIAA (CIAA 2024) will be held in Akita, Japan, from September 3rd to 6th, 2024. See `http://www.math.akita-u.ac.jp/ciaa2024/` for more information.

Workshop on Combinatorial Reconfiguration 2024
International workshop on combinatorial reconfiguration will be held in Fukuoka, Japan, from October 7th to 11th, 2024. This workshop will be the fifth in a series that was established in 2015, with events held in Japan, Canada (twice), and France. A major feature of this workshop series is to allocate ample time for participants to engage in working sessions tackling open problems, in addition to presenting the latest research findings related to combinatorial reconfiguration. See `https://joint.imi.kyushu-u.ac.jp/post-15540/` for more information.

ISAAC 2024
International Symposium on Algorithms and Computation (ISAAC) is intended to provide a forum for researchers working on algorithms and computation. The 35th edition of this symposium will be held in Sydney, Australia, from December 8th to 11th, 2024. See `https://sites.google.com/view/isaac2024/` for more information on ISAAC 2024.

Submission Deadline: June 28, 2024 (Anywhere on Earth)

Notification of Acceptance: September 2, 2024

Report on BCTCS 2024

The 40th British Colloquium for Theoretical Computer Science

4–5 April 2024, University of Bath

Thomas Powell

The British Colloquium for Theoretical Computer Science (BCTCS) is an annual forum in which researchers in Theoretical Computer Science can meet, present research findings, and discuss developments in the field. It also provides a welcoming environment for PhD students to gain experience in presenting their work to a broader audience, and to benefit from contact with established researchers.

BCTCS 2024 was hosted by the University of Bath and held from 4th to 5th April 2024, where it was co-located with the 5th meeting of the Southern and Midlands Logic Seminar. The event attracted 52 registered participants, and featured an interesting and wide-ranging programme. A total of 23 contributed talks – predominantly by PhD students – were presented at the meeting alongside the four keynote speakers. We are grateful for the sponsorship of the British Logic Colloquium which provided travel bursaries to a number of PhD students.

BCTCS 2024 was dedicated to the memory of Professor Alan Gibbons (1942-2024) who served as the second Chairman of the BCTCS from 1992 to 1998.

BCTCS 2025 will be hosted by the University of Strathclyde from 14rd–16th April 2025. Researchers and PhD students wishing to contribute talks concerning any aspect of Theoretical Computer Science are cordially invited to do so. Further details are available from the BCTCS website at `www.bctcs.ac.uk`.

Invited Talks

Anupam Das (University of Birmingham)
Proof theoretic approaches for regular languages (and beyond)

In the second half of the 20th century various theories of regular expressions were proposed, eventually giving rise to the notion of a Kleene Algebra (KA). Kozen and Krob independently proved the completeness of KA for the model of regular languages, a now celebrated result that has been refined and generalised over the years. In recent years proof theoretic approaches to regular languages have been studied, providing alternative routes to metalogical results like completeness and decidability. In this talk, I will present a new such approach from a different starting point: right-linear (or left-linear) grammars. The resulting ‘right-linear algebras' are more general than KAs, e.g. admitting ω-regular languages as a model too, and enjoy a simpler proof theory. This allows us to recover (more general) metalogical results in a robust way, combining techniques from cyclic proofs, games and automata.

Alex Kavvos (University of Bristol)
Two-dimensional Kripke Semantics

We revisit the duality between Kripke and algebraic semantics of intuitionistic and modal logic. We show that it is possible to raise this to the level of proofs, bridging the gap between Kripke semantics and more recent developments in modal type theory and the categorical semantics of modal logic.

Stuart Matthews (Capgemini Engineering)
Software engineering with formal methods: Quality, Time & Cost

Formal Methods – applying mathematics to software engineering – have seen a gradual industrial take-up since the early adopters in the 1980s. Early motivations were largely around questions of quality (Are we building the right system? Have we built the system right?) Until relatively recently, our assessment of the success of formal methods in becoming a mainstream industrial technique was relatively conservative, leading us to ask in a 2017 Royal Society paper, "Will the seedling ever flower?" In this talk I will give an update on this question, the answer now being rather more up-beat than it was seven years ago. We will look at the reasons for the increased penetration of formal methods, some of the companies that are leading the charge, and recent tool developments at Capgemini that are driving greater automation in the software development pipeline.

Monika Seisenberger (Swansea University)
Logic in Railway Verification

In this talk we present various logical methods we used for the verification of discrete and continuous Railway control systems. We start with an explanation of Ladder Logic, which is used in industrial control applications, present a semantics and show how a problem formulated in ladder logic amounts to a SAT solving problem. Then we talk about the modelling and verification of ERTMS (via modelchecking), and finally present more recent applications of SMT solving to the verification of Railway scheme plans. We conclude with a number of current challenges of formal methods in Railway.

Contributed Talks

Cezar-Mihail Alexandru (University of Bristol)
Interval Selection in Sliding Windows

We initiate the study of the Interval Selection problem in the (streaming) sliding window model of computation. We also establish a simple theoretical framework for proving memory lower bounds in this model which might be used beyond the Interval Selection problem. In this problem, an algorithm receives a potentially infinite stream of intervals on the line, and the objective is to maintain at every moment an approximation to a largest possible subset of disjoint intervals among the L most recent intervals, for some integer L. When processing streams, it is reasonable to focus on the most recent data items as it is modelled in the sliding window model since the near past usually affects the present more strongly than older data. In this talk, we will present the following results: In the unit-length intervals case, we give an optimal 2-approximation sliding window algorithm with space $O(|OPT|)$, and we briefly show that any sliding window algorithm that computes a $(2-\varepsilon)$-approximation must use space $\Omega(L)$ for any $\varepsilon > 0$. In the arbitrary-length case, we give a $(11/3 + \varepsilon)$-approximation sliding window algorithm with space $O(|OPT|)$, for any constant $\varepsilon > 0$. We also briefly show that space $\Omega(L)$ is needed for algorithms that compute a $(2.5 - \varepsilon)$-approximation, for any $\varepsilon > 0$. Our main technical contribution is an improvement over the smooth histogram technique, which consists of running independent copies of a traditional streaming algorithm with different start times. By employing the one-pass 2-approximation streaming algorithm by Cabello and Pérez-Lantero for Interval Selection on arbitrary-length intervals as the underlying algorithm, the smooth histogram technique immediately yields a $(4 + \varepsilon)$-approximation in this setting. Our improvement is obtained by forwarding the structure of the intervals identified in a run to the subsequent run, which constrains the "shape" of an optimal solution and allows us to target optimal intervals differently. This is based on joint work with Dr. Christian Konrad.

Pete Austin (University of Liverpool)
Parity Games on Temporal Graphs

Temporal graphs are a popular modelling mechanism for dynamic complex systems that extend ordinary graphs with discrete time. Simply put, time progresses one unit per step and the availability of edges can change with time. We consider the complexity of solving ω-regular games played on temporal graphs where the edge availability is ultimately periodic and fixed a priori. We show that solving parity games on temporal graphs is decidable in PSPACE, only assuming the edge predicate itself is in PSPACE. A matching lower bound already holds for what we call punctual reachability games on static graphs, where one player wants to reach the target at a given, binary encoded, point in time. We further study syntactic restrictions that imply more efficient procedures. In particular, if the edge predicate is in P and is monotonically increasing for one player and decreasing for the other, then the complexity of solving games is only polynomially increased compared to static graphs.

Hollie Baker (University of Bath)
Smooth stratification: an efficient implementation using SACLIB

A smooth stratification of an algebraic variety is a partition of that variety into effectively nonsingular subsets. Gabrielov and Vorobjov presented an algorithm for computing a basic weak stratification of a semi-Pfaffian set. I will present this algorithm along with a worked example and then describe its implementation on top of the computer algebra library SACLIB. Since SACLIB is a polynomial library, we will work in the category Semialgebraic Set, to which Gabrielov and Vorobjov's algorithm can be applied directly.

Karl Boddy (Newcastle University)
Bounded clique-width versus well-quasi-orderability by induced subgraphs

Boundedness of clique-width is a graph class property that has been extensively studied due its useful properties with respect to designing algorithms, with many NP problems tractable on graph classes with bounded clique-width. The graph class property of being well-quasi-ordered by the induced subgraph relation, at first seems completely unrelated when studying their definitions. However it has been conjectured by Lozin, Razgon and Zamarae that well-quasi-orderability implies boundedness of clique-width on finitely defined graph classes; that is classes which are defined by a finite number of forbidden induced subgraphs, which includes a large number of useful classes. As calculating clique-width in general is NP-hard, it would be useful to unite the two notions, expanding the number of classes we know to have bounded clique-width. We give an overview of the progress on this conjecture, including graph theoretical tools that are shared by the two properties. We focus on classes that are defined using only one or two forbidden graphs, where we have closed all but one case.

Adithya Diddapur (University of Bristol)
Interval Selection in Data Streams: Weighted Intervals and the Insertion-Deletion Setting

We study the problem in data streams: Given a stream of n intervals on the line, the objective is to compute a largest possible subset of non-overlapping intervals using $\tilde{O}(|OPT|)$ space, where $|OPT|$ is the size of an optimal solution. Previous work gave a $3/2$-approximation for unit-length and a 2-approximation for variable-length intervals, along with corresponding (tight) lower bounds. We extend this line of work to weighted intervals as well as to insertion-deletion streams. Our results include: When considering weighted intervals, a $(3/2 + \varepsilon)$-approximation can be achieved for unit intervals, but any constant factor approximation for variable length intervals requires space $\Omega(n)$. In the insertion-deletion setting where intervals can both be added and deleted, we prove that, even without weights, computing a constant factor approximation for variable length intervals requires space $\Omega(n)$, whereas in the weighted unit-length intervals case a $(2 + \varepsilon)$-approximation can be obtained. Our lower bound results are obtained via reductions to the recently introduced communication problem, further demonstrating the strength of this problem in the context of streaming geometric independent set problems. This talk is based on joint work with Jacques Dark and Christian Konrad.

Tala Eagling-Vose (Durham University)
Graph Homomorphisms, Colouring Games, and Forbidden Subgraphs

Recently, a framework has been proposed to investigate monotone graph classes, where a finite set of graphs is prohibited as subgraphs. In this talk, we explore both the framework and its impact through the lens of the graph homomorphism problem alongside some of its variants. A problem falls within our framework if it is efficiently solvable for graph classes of bounded treewidth, hard for the class of subcubic graphs, and remains hard when every edge is repeatedly subdivided. Under these conditions, a complete dichotomy is established for H-subgraph-free graphs, where H is any finite set of graphs. Graph homomorphism receives significant attention as it generalises the well-known graph colouring problem. In this talk, we consider a colouring game in which two players take turns colouring vertices of a graph. The first player wins if a proper colouring is obtained. Through this game, we demonstrate the presence of problems within the framework whose hardness is in the polynomial hierarchy above NP, in particular, a dichotomy exists between cases within P and those that are ΠP2k-complete.

Nathan Flaherty (University of Liverpool)
Collision-Free Robot Scheduling

Mobile robots are becoming an increasingly common part of scientific work within laboratory environments and therefore effectively coordinating them to complete necessary tasks around the laboratory whilst preventing collisions is of great practical interest. We model this theoretically by having a graph $G = (V, E)$ along with some tasks and robots, each task is on a vertex and has a certain duration, the time taken to complete the task, we then have robots which are agents that can move between adjacent vertices on the graph. The problem being to find schedules for the robots which complete all the tasks in G without any two robots sharing the same vertex or edge at any given time. We have shown that this is problem is NP-Complete even for some quite simple graph classes such as stars and cliques. This is work done jointly with Duncan Adamson, Igor Potapov and Paul Spirakis.

Iris van der Giessen (University of Birmingham)
Intuitionistic Gödel-Löb Logic, à la Simpson: Proof Theory and Birelational

Semantics

The Gödel-Löb's axiom is a modal axiom famous from the provability logic GL of Peano Arithmetic. The axiom represents a form of induction and is used in different constructive/intuitionistic settings to capture forms of recursion. We introduce a new Intuitionistic version of Gödel-Löb modal logic GL towards a computational interpretation of GL. While existing intuitionistic versions of GL are typically defined over only the box and not the diamond, we include both modalities. In the style of Alex Simpson, we develop a non-wellfounded labelled proof theory and coinciding birelational semantics and call the resulting logic IGL. This is joint work with Anupam Das and Sonia Marin.

Giulio Guerrieri (University of Sussex)
The theory of meaningfulness in the call-by-value lambda-calculus

The untyped lambda-calculus is a simple and Turing-complete model of computation that represents the kernel of any functional programming language. The semantics of the untyped call-by-name lambda-calculus is a well-developed field built around the concept of solvable terms, which are elegantly characterized in many different ways. In particular, unsolvable terms provide a consistent notion of meaningless terms, and meaningful terms can be identified with the solvable ones. The semantics of the untyped call-by-value lambda-calculus (CbV, which is closer to the real implementations of programming languages) is instead still in its early stages, because of some inherent difficulties but also because CbV solvable terms are less studied and understood than in call-by-name. On the one hand, we show that a carefully crafted presentation of CbV allows us to recover many of the properties that solvability has in call-by-name, in particular qualitative and quantitative characterizations via multi types. On the other hand, we stress that, in CbV, solvability plays a different role: identifying unsolvable terms as meaningless induces an inconsistent theory. We argue that in CbV, the correct notion of meaningful terms is captured by the concept of potential valuability. In particular, terms that are not potentially valuable provide a consistent notion of meaningless terms in CbV.

Thomas Karam (University of Oxford)
Fourier analysis modulo p on the Boolean cube

Fourier analysis in theoretical computer science is most commonly defined on the Boolean cube $\{0, 1\}^n$ identified with $\mathbb{Z}_2^n$. Recently, generalisations of that setting have been studied, involving restrictions to $\{0, 1\}^n$ of characters modulo p for some arbitrary prime p. Unlike in the case of mod-2 characters, different mod-p characters are no longer orthogonal when restricted to the Boolean cube. Equivalently, sets of mod-p linear forms which for $p = 2$ would be independent in a probabilistic sense provided that they are linearly independent, exhibit significantly more complicated behaviour for general p. We will begin by discussing some of the basic phenomena involved, and then briefly mention some recent progress and remaining difficulties.

David Kutner (Durham University)
Reconfigurable routing in data center networks

The Reconfigurable Routing Problem (RRP) in hybrid networks is, in short, the problem of finding settings for optical switches augmenting a static network so as to achieve optimal delivery of some given workload. The problem has previously been studied in various scenarios with both tractability and NP-hardness results obtained. However, the data center and interconnection networks to which the problem is most relevant are almost always such that the static network is highly structured, whereas all previous results assume that the static network can be arbitrary (which makes existing computational hardness results less technologically relevant and also easier to obtain). In this talk, we present results for restrictions of RRP where the underlying static network is highly structured, for example consisting of a hypercube.

Lukas Holter Melgaard (University of Birmingham)
Cyclic proofs for (arithmetical) inductive definitions

We investigate the cyclic proof theory of extensions of Peano Arithmetic by (finitely iterated) inductive definitions. Such theories are essential to proof theoretic analyses of certain 'impredicative' theories; moreover, our cyclic systems naturally subsume Simpson's Cyclic Arithmetic. Our main result is that cyclic and inductive systems for arithmetical inductive definitions are equally powerful. We conduct a metamathematical argument, formalising the soundness of cyclic proofs within second-order arithmetic and appealing to conservativity. This approach is inspired by those of Simpson and Das for Cyclic Arithmetic, however we must further address a difficulty that the closure ordinals of our inductive definitions (around Church-Kleene) far exceed the proof theoretic ordinal of the appropriate metatheory (around Bachmann-Howard or Bucholz), and so explicit induction on their notations is not possible. For this reason, we rather rely on a formalisation of the theory of (recursive) ordinals within second-order arithmetic.

Kheeran Naidu (University of Bristol)
Multi-Pass Streaming Lower Bounds for Graph Problems

Many one-pass graph streaming lower bounds (e.g., connectivity, maximum matching, maximal independent set) are proven by a reduction to the classic one-round one-way two-party communication problem INDEX, which is hard in one round. In two rounds, however, it becomes exponentially easier and is thus not suitable for proving non-trivial multi-pass lower bounds. In this talk, we look at the communication game HiddenStrings introduced by Konrad and Naidu, which can be seen as a generalisation of INDEX, and show that it is suitably hard in multiple rounds/passes. In particular, we can prove multi-pass semi-streaming lower bounds for Approximate Maximum Matching and Maximal Independent Set (by Assadi et al.). This is based on joint works with Sepehr Assadi, Christian Konrad, and Janani Sundaresan.

Ian Price (Swansea University)
Two-Way Reversible Transducers as Functors

Hines suggested a notion of planarity for two-way deterministic finite automata, conjecturing links between automata theory and Temperley-Lieb algebras familiar from Physics and Knot theory. Taking inspiration from this, Pradic and Nguyên introduced a notion of planarity for two-way finite reversible automata (2RFAs) and transducers (2RFTs), then showed that these compute star-free languages and first-order transductions, respectively. Building on this work, I will explain how to view 2RFTs as a functor from a particular "automata shape" category, due to Colcombet & Petrisan, to a category of planar transition diagrams. This category is autonomous (like compact-closed, but not symmetric) and can be viewed as a target for interpreting non-commutative lambda calculi. I will work through a particular example and hint at future applications to linear logic.

Tymofii Prokopenko (University of Liverpool)
Capturing an invisible robber on a planar graph

"Cops and robbers" is a game on a graph where a team of cops starting at a given vertex on a given graph (known as an arena) aims to capture a robber starting at another vertex (i.e. at least one of the team of cops must be in the same vertex as the robber). Moves alternate between a team of cops and a robber and in classical settings all players move along the edges of a given graph. Two main variants of a game with respect to the visibility of a robber by a team of cops: full visibility and zero visibility. In full visibility case the cops know the location of the robber on the graph and thus have full information at any time. In the zero visibility case, the robber is invisible to the cops throughout the game and only at the moment of capturing the robber the location becomes known. We present an algorithm for capturing the robber in the case of zero-visibility by a team of robots on a planar graph which extends the known results for simpler structures like trees. Following the formal analysis, we estimate the number of cops and the time/space complexity required for capturing the robber. Moreover, we define the new problem for a game with a mixed setting of full and zero visibility of a robber.

Chris Purdy (Royal Holloway, University of London)

A cyclical proof system for higher-order fixed point logic

Non-well-founded proof systems, allowing infinitely deep derivations, facilitate implicit forms of (co)inductive reasoning, validated by a so-called global trace condition requiring each infinite path in the derivation to witness some infinite descent. Reasoning using cyclic proofs, the finite representations of regular such non-well-founded derivations, is often more natural than using induction rules, since explicit induction invariants (hypotheses) are not required. We present work-in-progress on a non-well-founded (cyclic) natural deduction system for higher order fixed point logic with explicit approximations muHOLex. To our knowledge, ours is the first cyclic proof system to combine fixed point operators and negation in a higher order setting. We also discuss the development of a suitable global trace condition in the absence of explicit approximations, as well as potential propositions-as-types interpretations for both of these deduction systems.

Andrew Ryzhikov (University of Oxford)
Fixed Vector Addition Systems and Bounded Arithmetic

Recently, there has been significant success in understanding the computational complexity of the reachability problems in vector addition systems with states (VASS). We consider a variant of this problem where the VASS is fixed, and only its initial and target configurations are a part of the input. If the input is given in the binary encoding, we show that there exists a VASS such that this problem is PSPACE-hard. If the input is given in the unary encoding, the problem becomes more difficult to analyse, since in this case the input consists of a constant-length sequence of unary numbers. Despite that, VASS remain quite expressive: in particular, for every fixed formula of bounded arithmetic with counting there exists a fixed VASS simulating this formula in some natural way. This is a joint work with Christoph Haase and Andrei Draghici.

James Swire (Swansea University)
An empirical evaluation of a quality assurance process: A case study in the railway sector

Critical systems, such as railways, require multiple levels of testing and verification to be fit for use. The objective of this research is to determine the quality of a verification approach for the control software of railway interlocking computers. As such, we will utilise statistics to determine the likelihood of a fault being detected on any given track plan using its respective safety properties. To accomplish this, we will inject faults into an otherwise correct control software and see if the verification approach detects them. A secondary aim of this research is to automate this fault injection process. Our research aims to establish a fault probability space, determined by these statistics, of control software after passing the verification process.

Marc Thatcher (University of Sussex)
Parallel Functional Programming with Interaction Nets

Interaction nets are a graph-rewriting system based on Girard's linear logic. Suitable as both a computational model and a simple programming language, a most interesting aspect of the latter case is the ease with which they can move from sequential to parallel execution. The combination of locality (graph rewrites are defined on pairs of nodes which do not affect the greater graph) and linearity (any sub-graph's interface with the greater graph must be unchanged by the reduction rules) means that nets can automatically take advantage of any parallelism in the algorithm, without any extra work from the programmer. Interaction net programming languages proposed to date have used various forms of explicit syntax, which is easy to reason about but hard to use. We propose a functional-like programming language for defining interaction nets with a bijective mapping from the language to nets and show how this can be used to provide parallel programming "for free". An alpha-version of an implementation of this language is available.

Riccardo Treglia (Kings College London)
Monadic Intersection Types, Relationally

We consider an extension of intersection types to a computational lambda-calculus with algebraic operations à la Plotkin and Power. We achieve this by considering monadic intersections – whereby computational effects appear not only in the operational semantics, but also in the type system. We will focus on why in the effectful setting termination is not anymore the only property of interest, and consequently how to analyze the interactive behaviour of typed programs with the environment is of interest. As the main result, our type system is able to characterize the natural notion of observation, both in the finite and in the infinitary setting, and for a wide class of effects. This is achieved via a novel combination of syntactic techniques with abstract relational reasoning, which we will briefly introduce as a tool to lift all the required notions to the monadic setting. This is joint work with Francesco Gavazzo and Gabriele Vanoni.

Sean Watters (University of Strathclyde)
Extensional Finite Sets and Multisets in Agda

Lists and list-like types are easy to represent in type theory. However, sets and multisets are more troublesome; without advanced features such as quotient types, representations of these typically lack either decidable equality or the correct equational theory. In this talk, I will discuss how to realise finite sets and multisets in Agda, while maintaining the above desiderata. We achieve this using a generalisation of Catarina Coquand's data type of fresh lists. Our constructions satisfy properties akin to (multi)set extensionality, which allows us to prove that they satisfy the universal properties of the free idempotent commutative monoid and free commutative monoid, respectively. I will also show how fresh lists realise many other free algebraic structures. This is joint work with Clemens Kupke and Fredrik Nordvall Forsberg.

Xin Ye (Durham University)
Computing Balanced Solutions for Large International Kidney Exchange Schemes When Cycle Length Is Unbounded

In kidney exchange programmes (KEP) patients may swap their incompatible donors leading to cycles of kidney transplants. Nowadays, countries try to merge their national patient-donor pools leading to international KEPs (IKEPs). As shown in the literature, long-term stability of an IKEP can be achieved through a credit-based system. In each round, every country is prescribed a "fair" initial allocation of kidney transplants. The initial allocation, which we obtain by using solution concepts from cooperative game theory, is adjusted by incorporating credits from the previous round, yielding the target allocation. The goal is to find, in each round, an optimal solution that closely approximates this target allocation. There is a known polynomial-time algorithm for finding an optimal solution that lexicographically minimizes the country deviations from the target allocation if only 2-cycles (matchings) are permitted. In practice, kidney swaps along longer cycles may be performed. However, the problem of computing optimal solutions for maximum cycle length ℓ is NP-hard for every $\ell \geq 3$. This situation changes back to polynomial time once we allow unbounded cycle length. However, in contrast to the case where $\ell = 2$, we show that for $\ell = \infty$, lexicographical minimization is only polynomial-time solvable under additional conditions (assuming sP≠NP). Nevertheless, the fact that the optimal solutions themselves can be computed in polynomial time if $\ell = \infty$ still enables us to perform a large scale experimental study for showing how stability and total social welfare are affected when we set $\ell = \infty$ instead of $\ell = 2$.

Tansholpan Zhanabekova (University of Liverpool)
Semantic Flowers for Good-for-Games and Deterministic Automata

We present an innovative approach for capturing the complexity of ω-regular languages using the concept of flowers. This semantic tool combines two syntax-based definitions, namely the Mostowski hierarchy of word languages and syntactic flowers. The former is based on deterministic parity automata with a limited number of priorities, while the latter simplifies deterministic parity automata by reducing the number of priorities used, without altering their structure. Synthesising these two approaches yields a semantic concept of flowers, which offers a more effective way of dealing with the complexity of ω-regular languages. This latter provides a comprehensive definition of semantic flowers and shows that it captures the complexity of ω-regular languages. We also show that this natural concept yields simple proofs of the expressive power of good-for-games automata.

The Bulletin of the EATCS

European Association for Theoretical Computer Science

E A T C S

105

EATCS

HISTORY AND ORGANIZATION

EATCS is an international organization founded in 1972. Its aim is to facilitate the exchange of ideas and results among theoretical computer scientists as well as to stimulate cooperation between the theoretical and the practical community in computer science.

Its activities are coordinated by the Council of EATCS, which elects a President, Vice Presidents, and a Treasurer. Policy guidelines are determined by the Council and the General Assembly of EATCS. This assembly is scheduled to take place during the annual **I**nternational **C**olloquium on **A**utomata, **L**anguages and **P**rogramming (ICALP), the conference of EATCS.

MAJOR ACTIVITIES OF EATCS

- Organization of ICALP;
- Publication of the "Bulletin of the EATCS;"
- Award of research and academic career prizes, including the EATCS Award, the Gödel Prize (with SIGACT), the Presburger Award, the EATCS Distinguished Dissertation Award, the Nerode Prize (joint with IPEC) and best papers awards at several top conferences;
- Active involvement in publications generally within theoretical computer science.

Other activities of EATCS include the sponsorship or the cooperation in the organization of various more specialized meetings in theoretical computer science. Among such meetings are: CIAC (Conference of Algorithms and Complexity), CiE (Conference of Computer Science Models of Computation in Context), DISC (International Symposium on Distributed Computing), DLT (International Conference on Developments in Language Theory), ESA (European Symposium on Algorithms), ETAPS (The European Joint Conferences on Theory and Practice of Software), LICS (Logic in Computer Science), MFCS (Mathematical Foundations of Computer Science), WADS (Algorithms and Data Structures Symposium), WoLLIC (Workshop on Logic, Language, Information and Computation), WORDS (International Conference on Words).

Benefits offered by EATCS include:
- Subscription to the "Bulletin of the EATCS;"
- Access to the Springer Reading Room;
- Reduced registration fees at various conferences;
- Reciprocity agreements with other organizations;
- 25% discount when purchasing ICALP proceedings;
- 25% discount in purchasing books from "EATCS Monographs" and "EATCS Texts;"
- Discount (about 70%) per individual annual subscription to "Theoretical Computer Science;"
- Discount (about 70%) per individual annual subscription to "Fundamenta Informaticae."

Benefits offered by EATCS to Young Researchers also include:
- Database for Phd/MSc thesis
- Job search/announcements at Young Researchers area

(1) THE ICALP CONFERENCE

ICALP is an international conference covering all aspects of theoretical computer science and now customarily taking place during the second or third week of July. Typical topics discussed during recent ICALP conferences are: computability, automata theory, formal language theory, analysis of algorithms, computational complexity, mathematical aspects of programming language definition, logic and semantics of programming languages, foundations of logic programming, theorem proving, software specification, computational geometry, data types and data structures, theory of data bases and knowledge based systems, data security, cryptography, VLSI structures, parallel and distributed computing, models of concurrency and robotics.

Sites of ICALP meetings:

- Paris, France 1972
- Saarbrücken, Germany 1974
- Edinburgh, UK 1976
- Turku, Finland 1977
- Udine, Italy 1978
- Graz, Austria 1979
- Noordwijkerhout, The Netherlands 1980
- Haifa, Israel 1981
- Aarhus, Denmark 1982
- Barcelona, Spain 1983
- Antwerp, Belgium 1984
- Nafplion, Greece 1985
- Rennes, France 1986
- Karlsruhe, Germany 1987
- Tampere, Finland 1988
- Stresa, Italy 1989
- Warwick, UK 1990
- Madrid, Spain 1991
- Wien, Austria 1992
- Lund, Sweden 1993
- Jerusalem, Israel 1994
- Szeged, Hungary 1995
- Paderborn, Germany 1996
- Bologne, Italy 1997
- Aalborg, Denmark 1998
- Prague, Czech Republic 1999
- Genève, Switzerland 2000
- Heraklion, Greece 2001
- Malaga, Spain 2002
- Eindhoven, The Netherlands 2003
- Turku, Finland 2004
- Lisabon, Portugal 2005
- Venezia, Italy 2006
- Wrocław, Poland 2007
- Reykjavik, Iceland 2008
- Rhodes, Greece 2009
- Bordeaux, France 2010
- Zürich, Switzerland 2011
- Warwick, UK 2012
- Riga, Latvia 2013
- Copenhagen, Denmark 2014
- Kyoto, Japan 2015
- Rome, Italy 2016
- Warsaw, Poland 2017
- Prague, Czech Republic 2018
- Patras, Greece 2019
- Saarbrücken, Germany (virtual conference) 2020
- Glasgow, UK (virtual conference) 2021
- Paris, France 2022
- Paderborn, Germany 2023

(2) THE BULLETIN OF THE EATCS

Three issues of the Bulletin are published annually, in February, June and October respectively. The Bulletin is a medium for *rapid* publication and wide distribution of material such as:

- EATCS matters;
- Technical contributions;
- Columns;
- Surveys and tutorials;
- Reports on conferences;
- Information about the current ICALP;
- Reports on computer science departments and institutes;
- Open problems and solutions;
- Abstracts of Ph.D. theses;
- Entertainments and pictures related to computer science.

Contributions to any of the above areas are solicited, in electronic form only according to formats, deadlines and submissions procedures illustrated at `http://www.eatcs.org/bulletin`. Questions and proposals can be addressed to the Editor by email at `bulletin@eatcs.org`.

(3) OTHER PUBLICATIONS

EATCS has played a major role in establishing what today are some of the most prestigious publication within theoretical computer science.

These include the *EATCS Texts* and the *EATCS Monographs* published by Springer-Verlag and launched during ICALP in 1984. The Springer series include *monographs* covering all areas of theoretical computer science, and aimed at the research community and graduate students, as well as *texts* intended mostly for the graduate level, where an undergraduate background in computer science is typically assumed.

Updated information about the series can be obtained from the publisher.

The editors of the EATCS Monographs and Texts are now M. Henzinger (Vienna), J. Hromkovič (Zürich), M. Nielsen (Aarhus), G. Rozenberg (Leiden), A. Salomaa (Turku). Potential authors should contact one of the editors.

EATCS members can purchase books from the series with 25% discount. Order should be sent to:
Prof.Dr. G. Rozenberg, LIACS, University of Leiden,
P.O. Box 9512, 2300 RA Leiden, The Netherlands
who acknowledges EATCS membership and forwards the order to Springer-Verlag.

The journal *Theoretical Computer Science*, founded in 1975 on the initiative of EATCS, is published by Elsevier Science Publishers. Its contents are mathematical and abstract in spirit, but it derives its motivation from practical and everyday computation. Its aim is to understand the nature of computation and, as a consequence of this understanding, provide more efficient methodologies.

The Editor-in-Chief of the journal currently are D. Sannella (Edinburgh), L. Kari and P.G. Spirakis (Patras).

ADDITIONAL EATCS INFORMATION

For further information please visit `http://www.eatcs.org`, or contact the President of EATCS:
Prof. Artur Czumaj,
Email: `president@eatcs.org`

EATCS MEMBERSHIP

DUES

The dues are €40 for a period of one year (two years for students / Young Researchers). Young Researchers, after paying, have to contact `secretary@eatcs.org`, in order to get additional years. A new membership starts upon registration of the payment. Memberships can always be prolonged for one or more years.

In order to encourage double registration, we are offering a discount for SIGACT members, who can join EATCS for €35 per year. We also offer a five-euro discount on the EATCS membership fee to those who register both to the EATCS and to one of its chapters. Additional €35 fee is required for ensuring the *air mail* delivery of the EATCS Bulletin outside Europe.

HOW TO JOIN EATCS

You are strongly encouraged to join (or prolong your membership) directly from the EATCS website `www.eatcs.org`, where you will find an online registration form and the possibility of secure online payment. Alternatively, contact the Secretary Office of EATCS:

Mrs. Efi Chita,
Computer Technology Institute & Press (CTI)
1 N. Kazantzaki Str., University of Patras campus,
26504, Rio, Greece
Email: `secretary@eatcs.org`,
Tel: +30 2610 960333, Fax: +30 2610 960490

If you are an EATCS member and you wish to prolong your membership or renew the subscription you have to use the Renew Subscription form. The dues can be paid via paypal and all major credit cards are accepted.

For adittional information please contact the Secretary of EATCS:

Dmitry Chistikov
Computer Science
University of Warwick
Coventry
CV4 7AL
United Kingdom
Email: `secretary@eatcs.org`,